Adabas

DBA Tasks

Version 8.6.2

October 2026

ADABAS & NATURAL

This document applies to Adabas Version 8.6.2 and all subsequent releases.

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

Copyright © 1971-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

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

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

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

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

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

Document ID: ADAMF-DBATASKS-862-20261013

Table of Contents

Preface vii
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 DBA Roles and Responsibilities 5
Central Control and Coordination 6
The DBA in the IS Organization 7
Establishing Database Control and Administration 9
Data Definition and Control 13
Database Documentation 16
Education and Training 24
The DBA and the User 27
The DBA and Application Selection/Development 30
The DBA and Computer Operations 35
3 Database Design 41
Performance Control During System Design 42
File and Record Design 44
Data Access Strategies 50
Disk Space Usage 56
Adabas Security 60
Recovery/Restart Design 63
The Adabas Recovery Aid 69
Multiclient Support 70
4 Date-Time Edit Mask Reference 79
E(DATE) Edit Mask 80
E(TIME) Edit Mask 80
E(DATETIME) Edit Mask 81
E(TIMESTAMP) Edit Mask 81
E(NATDATE) Edit Mask 82
E(NATTIME) Edit Mask 83
E(UNIXTIME) Edit Mask 84
E(XTIMESTAMP) Edit Mask 84
Conversions Between Date-Time Representations 85
Conversions Between Date-Time Representations for Fields with the TZ option 88
Default and Null Value Handling 90
Value Verification 91
5 Expanded Files 93
Defining Expanded Files 94
Inserting a Component File 96
Removing a Component File 96
Deleting Expanded Files 97

Inspecting an Expanded File ... 97
Expanded Files and the Adabas Nucleus ... 97
Expanded Files and Adabas Utilities ... 98
6 Large Object (LB) Files and Fields ... 103
Getting Started with Large Object (LB) Fields ... 104
LOB File Management ... 107
Processing LOB Field Segments ... 107
7 Defining an Adabas Database ... 115
Step 1 : Estimate the Size of the Database ... 116
Step 2 : Allocate Space ... 134
Step 3 : Format the Space ... 136
Step 4 : Define Database Parameters ... 136
8 Database Space Management ... 137
Adabas Physical Extents ... 138
Adabas Device Type ... 138
Relative Adabas Block Number (RABN) ... 139
Adabas Logical Extents ... 140
Adabas Space Allocation and Deallocation ... 141
Using the Database Status Report to Control Space Use ... 150
Potential Space Use Problems and Recommended Actions ... 150
Receiving Associator and Data Storage Space Usage Notifications ... 152
Increasing Associator and Data Storage Space ... 153
Secondary Space Considerations when Defining and Extending Adabas Data Sets ... 156
9 Database Monitoring and Tuning ... 157
Monitoring Resource Use ... 158
Reporting on Resource Use ... 158
Monitoring Database Controls ... 158
Performance Management, Statistics, and Tuning ... 159
Adabas Session Statistics ... 160
Command Logging ... 168
10 Error Handling and Message Buffering ... 169
User Exit Failures ... 170
Recovery or Plug-In (PIN) Routines ... 171
PIN Routine User Exit ... 181
11 Universal Encoding Support (UES) ... 185
Wide-Character Encodings ... 186
Wide-Character Data Support ... 188
12 Multiple Platform Support ... 191
Encodings ... 192
ADACOX Conversion Exit ... 193
Conversion of High Value in Value Buffer ... 193
Data Translation Restrictions ... 194
Platform Considerations ... 195
13 Adabas SMF Records ... 197

Record Structure 198
Record Size Limits 198
Record Subtypes 199
Statistical Recording 200
Record Sections 200
ASMFREC Mapping Macro 217
SMF User Exit 218
IBM Type 89 SMF Records 219
14 AFPLOOK 221
Enabling AFPLOOK 222
Operational Defaults 222
Adjusting the Defaults 222
AFPLOOK Parameters 223
AFPLOOK Report 225
15 AVILOOK 229
Enabling AVILOOK 230
SYSAVI – Selecting AVILOOK 230
SYSAVI – Using AVILOOK 231
16 Adabas Online System Demo Version 235
Overview 236
Main Menu Functions 238
Session Monitoring 240
List Checkpoints 252
File Maintenance 255
Database Maintenance 257
System Operator Command Functions 258
Database Report 265
Index 273

Preface

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	Reviews the role and functions of the database administrator (DBA).
Database Design	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, and multiclient files.
Date-Time Edit Mask Reference	Provides information on each of the possible date-time edit masks and processing and conversion information for them.
Expanded Files	Provides information about expanded files.
Large Object (LB) Files and Fields	Provides information about large object (LOB) files and fields.
Defining an Adabas Database	Describes the procedure for defining an Adabas database.
Database Space Management	Provides pertinent information related to database space management.
Database Monitoring and Tuning	Provides information about monitoring and tuning your Adabas database.
Error Handling and Message Buffering	Describes Adabas error handling and message buffering.
Universal Encoding Support (UES)	Describes universal encoding support (UES) for Adabas databases.
Multiple Platform Support	Describes Adabas data translation for multiple platform support.
Adabas SMF Records	Describes Adabas SMF records and how they can be produced.
Adabas Online System Demo Version	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	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	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.

1 About this Documentation

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

Document Conventions

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

Online Information and Support

Product Documentation

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

Product Training

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

Tech Community

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

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

Product Support

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

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

Data Protection

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

2 DBA Roles and Responsibilities

- Central Control and Coordination 6
- The DBA in the IS Organization 7
- Establishing Database Control and Administration 9
- Data Definition and Control 13
- Database Documentation 16
- Education and Training 24
- The DBA and the User 27
- The DBA and Application Selection/Development 30
- The DBA and Computer Operations 35

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.

In the context of this documentation, the DBA 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.

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.

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.

- Position of the DBA in the Organization
- Necessary Attributes for a DBA
- Management Support
- What Mistakes Are Possible?

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.

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.

- Establishing Database Procedures and Standards
- Maintaining Procedures and Standards
- Assisting in Database Design
- Educating Users
- Selecting Applications Suitable for the Database System
- DBA Function Summary

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 some rules 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 data sets in the Adabas environment might require the following:

  ```
  Using Temporary Data Sets
  To ensure that your job is recovery-/restart-capable, always catalog temporary ↩
  data sets.
  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 data sets ↩
  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

Data Definition and Control

This section provides an overview for database administrators in regards to managing their Adabas databases.

- Planned Approach: Central Control of Data
- Determining Responsibility for Data
- Selecting Applications: Advising on System Development
- Advising on Data Collection and Validation
- Defining Database Contents

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	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	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.

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.

- Standards
- Description of the Database
- Data Dictionary, Function and Use
- Predict: The Adabas Data Dictionary
- Applications Using the Database
- Description of Data Sources
- Data Access and Manipulation Procedures
- Passwords and User Identification
- Backup Procedures
- Restart and Recovery Procedures for Database Documentation
- DBMS Performance and Measurement

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, key holders, 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.

Backup Procedures

The content of backup 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 for Database Documentation

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.

Education and Training

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.

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

- Database Concepts
- Database Design
- Programming
- Operating Procedures and Techniques
- Data Entry

- Database Query and Report Generation

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 GmbH, 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.

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.

- Liaison with the User
- Access Requirements
- Application Interface
- Complying with Standards and Controls

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 primary 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 us. 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.

The DBA and Application Selection/Development

This section provides an overview of application selection and planning for your database.

- Configuration and Applications Planning
- Database Organization
- Understanding Current and Future User Requirements
- Coordinating Database Activities
- Analyzing Access Requirements
- Establishing Data Availability
- Performance Versus Flexibility
- Advising on Application/Program/Database Design
- Determining Physical Storage Requirements
- The Test Database and Testing Strategy

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 GmbH 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 appreciate and be aware of the user's current and future requirements. In this instance, *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 GmbH, 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.

The DBA and Computer Operations

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.

This section provides an overview on the job of a database administrator in the context of computer operations.

- Scheduling Computer Time
- Operating Procedures
- Restart and Recovery Procedures for Computer Operations
- Database Utilities

- Working with Software GmbH

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 for Computer Operations

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 GmbH

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

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 GmbH. Such a request should always come from the DBA.

Software GmbH 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 GmbH. 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 GmbH. Nonetheless, Software GmbH 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 GmbH'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 GmbH.

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. We encourage 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.

3 Database Design

- Performance Control During System Design 42
- File and Record Design 44
- Data Access Strategies 50
- Disk Space Usage 56
- Adabas Security 60
- Recovery/Restart Design 63
- The Adabas Recovery Aid 69
- Multiclient Support 70

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.

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.

- Methodology for Performance Control in System Design

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.

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 queuing 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).

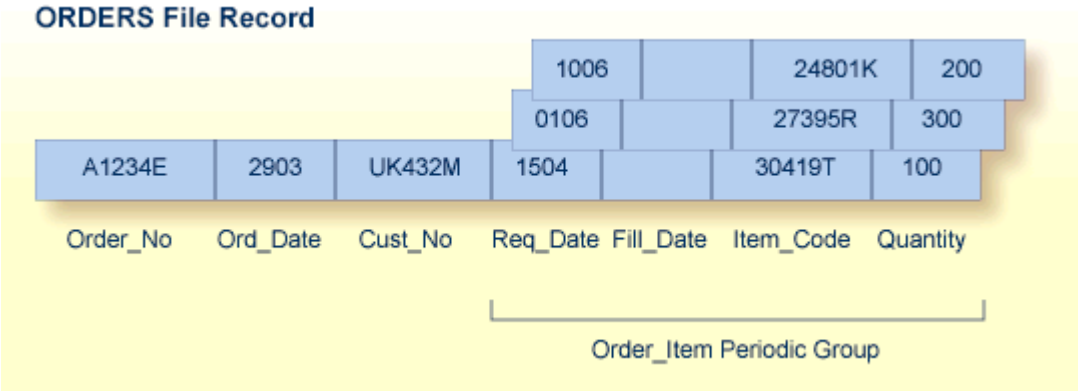
Each of these techniques is described in the following sections.

- Multiple-Value Fields and Periodic Groups
- Different Record Types in a Single Adabas File
- Linking Physical Files in a Single Logical File
- Data Duplication
- Adabas Record Design

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



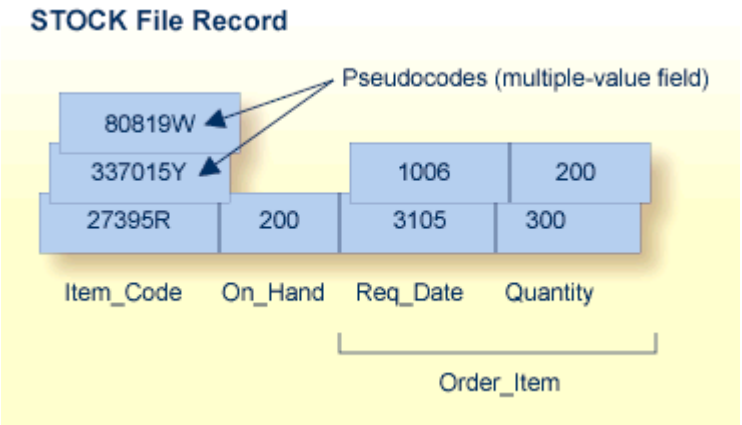
In the example shown in the table above, the order information in the table is shown converted to a record format in an Adabas file called ORDERS. Each order record contains a periodic group to permit a variable number of order items. In this case, the periodic group ORDER_ITEM, comprising the ITEM_CODE field (order item code) and the related fields QUANTITY (quantity desired), REQ_DATE (date required), and FILL_DATE (actual date the order was filled), can specify up to 65534 different items in a given record. Each unique occurrence of the ORDER_ITEM periodic group is called an *occurrence*; up to 65534 occurrences per periodic group are possible.

Note: The use of more than 191 MU fields or PE groups in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS `MUPEX` and `MUPECOUNT` parameters.

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).



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 labeled 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 65534 values, depending on the settings for the file. 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 (MU) field can be up to 65534. The actual number of occurrences allowed must be explicitly set for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS `MUPEX` and `MUPECOUNT` parameters.
- The maximum number of occurrences of any one periodic group (PE) can be up to 65534. The actual number of occurrences allowed must be explicitly set for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS `MUPEX` and `MUPECOUNT` parameters.
- 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 three-byte ISNs can contain a maximum of 16777215 records; a file with four-byte ISNs can contain 4294967294 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 4294967294 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

Data duplication can be part of database design in either of two ways:

- **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 totaling 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 COMPRESS 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.

Data Access Strategies

This section describes various data access strategies available in Adabas.

- Efficient Use of Descriptors
- Collation Descriptor
- Superdescriptor
- Subdescriptor
- Phonetic Descriptor
- Hyperdescriptor
- File Coupling

- User-Assigned ISNs
- Using the ISN as a Descriptor
- ADAM Usage

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. Superdescriptors 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.

For complete information about defining superdescriptors, read *SUPDE: Superdescriptor Definition* in the ADACMP documentation found in *Adabas Utilities Manual*.

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 hyperdescriptor exits, 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
- Logical Coupling

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.

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.

- Data Compression
- Forward Index Compression
- Padding Factors

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 (for example, *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
- Ordinary Compression
- Null-Value Suppression

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

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.

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.

- Security Planning
- Password Security
- Security by Value
- Ciphering
- Using Adabas SAF Security
- Natural and Adabas Online System Security

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.

Using Adabas SAF Security

Adabas SAF Security, an Adabas add-on product, can be used with Software GmbH's Com-plete and with the following non-Software GmbH security environments:

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

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

Natural and Adabas 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.

Recovery/Restart Design

This section discusses the design aspects of database recovery and restart. Proper recovery and 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.

- Adabas Recovery
- Planning and Incorporating Recoverability
- Matching Requirements and Facilities
- Transaction Recovery
- End Transaction (ET) Command
- Close (CL) Command
- Reading ET Data
- System or Transaction Failure
- Limitations of Adabas Transaction Recovery
- Adabas Checkpoint Commands
- Exclusive File Control

- User Restart Data

Adabas Recovery

Recovery of database integrity has the highest priority; if a database transaction fails or must be canceled, 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.

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.

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*).

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 determine 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.

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 data set.

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.

- The Recovery Log (RLOG)
- Starting the Recovery Aid

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/RESTORE database or RESTORE GCB function is executed following ADARAI PREPARE.

All activity between the first and second ADASAV SAVE/RESTORE database 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.

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.

- The Owner Concept
- Superusers
- Program Compatibility
- Support for Soft Coupling
- Data and Index Structures
- Performance Considerations
- User Profile Table

- Possible Adabas Response Codes
- Utility Support for Multiclient Files

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. Here, USER1, USER2 and USER3 share the same owner ID and therefore the same records.
USER2	1	
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	

Superusers

A *superuser* owner ID provides access to all records in a multiclient file. A superuser 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 superuser owner IDs, each beginning with an asterisk and allowing identical privileges, can be defined for a multiclient file.

A superuser owner ID applies only to L*x* read commands and nondescriptor search (S*x*) commands. Descriptor search commands by a superuser return only the records having the superuser's owner ID. Data records or index values stored by a superuser are labeled with the superuser's owner ID.

Note: If a superuser 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.

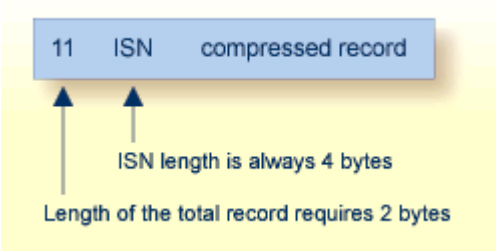
Data and Index Structures

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

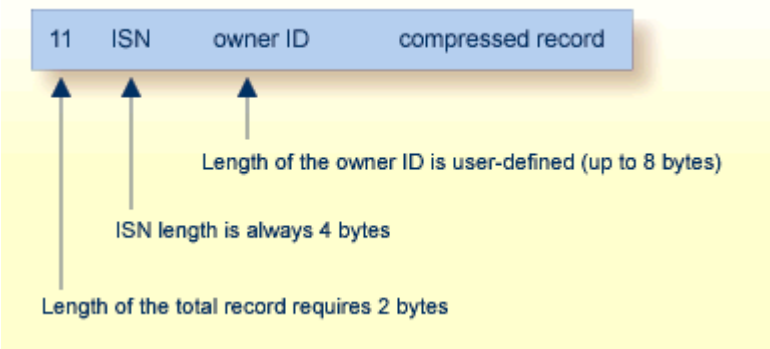
- Data Storage
- Associator

Data Storage

A Data Storage (DATA) record in a standard file has the following structure:

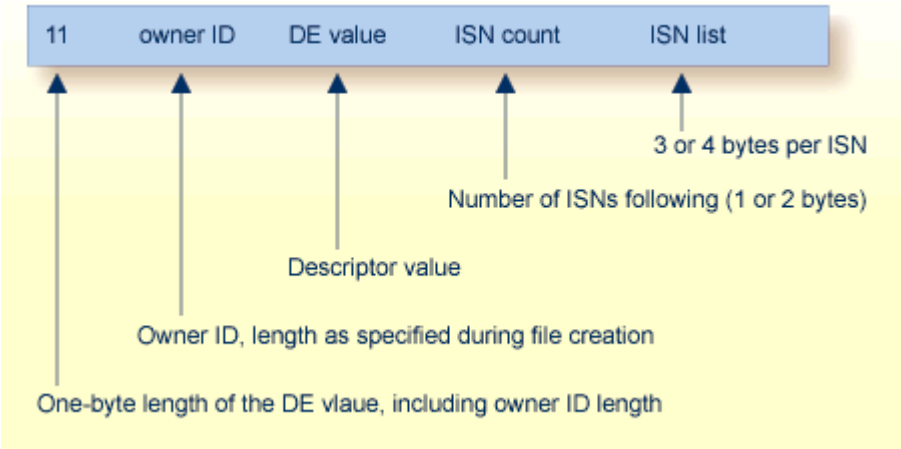


A Data Storage record in a multiclient file has the following structure:



Associator

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



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 superuser 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 (ADARSP003) or 113 (ADARSP113), 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 (ADARSP068) 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 (ADARSP003) or 113 (ADARSP113) 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
- The ADALOD Utility UPDATE Function
- The ADAULD Utility
- The ADACMP Utility

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 non-multiclient file	Convert into a non-multiclient 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 non-multiclient 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 overwritten 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.

4 Date-Time Edit Mask Reference

- E(DATE) Edit Mask 80
- E(TIME) Edit Mask 80
- E(DATETIME) Edit Mask 81
- E(TIMESTAMP) Edit Mask 81
- E(NATDATE) Edit Mask 82
- E(NATTIME) Edit Mask 83
- E(UNIXTIME) Edit Mask 84
- E(XTIMESTAMP) Edit Mask 84
- Conversions Between Date-Time Representations 85
- Conversions Between Date-Time Representations for Fields with the TZ option 88
- Default and Null Value Handling 90
- Value Verification 91

This chapter describes each of the possible date-time edit masks and provides processing and conversion information for them. It covers the following topics:

E(DATE) Edit Mask

Fields with the E(DATE) format are composed of eight digits in YYYYMMDD format, where YYYY can range from "0001" through "9999", MM can range from "01" through "12", and DD can range from "01" through "31". DATE must be valid according to the Gregorian calendar -- as long as there is no support for other calendars. This means that February 29th (Leap day) is only valid on Leap years (years divisible by four except when divisible by 100 but not divisible by 400).

Historically, the Gregorian calendar started to replace the Julian calendar on October 15, 1582 (the cutover date). When DATE format is used, earlier dates than October 15, 1582 are handled according to the same Gregorian calendar rules (as if the Gregorian calendar was in effect before the cutover date).

Negative values are not allowed and empty values are represented as zero.

Formats	Dec. 31, 2008	Empty Value	Jan 1,1 (minimum)	Dec. 31, 9999 (maximum)
U.8	Z'20081231'	Z'00000000'	Z'00010101'	Z'99991231'
P,5	X'020081231F'	X'000000000F'	X'000010101F'	X'099991231F'
B,4 and F,4	F'20081231'	F'0'	F'10101'	F'99991231'

- Unpacked format is the concatenation of year, month, and day, written as character digits. This is also called "zoned form" (e.g., '0' is X'F0' in EBCDIC or X'30' in ASCII).
- Packed data is the unpacked value packed.
- Binary or fixed point format is calculated as: `year*10000 + month*100 + day*1`.

E(TIME) Edit Mask

Fields with the E(TIME) format are composed of six digits in HHMMSS format, where HH can range from "00" through "23" and MM and SS can each range from "00" through "59". Leap seconds (SS=60) are not supported. Negative times are not allowed.

Formats	13:20:59 (sample)	Empty Value (default)	00:00:00 (minimum)	23:59:59 (maximum)
U.6	Z'132059'	Z'000000'	Z'000000'	Z'235959'
P,4	X'0132059F'	X'0000000F'	X'0000000F'	X'0235959F'
B,3 and F,4	F'132059'	F'0'	F'0'	F'235959'

- Unpacked format is the concatenation of hours + minutes + seconds
- Packed data is the unpacked value packed.
- Binary or fixed point format is calculated as the binary number of: `hour*10000 + minutes*100 + seconds*1`.

E(DATETIME) Edit Mask

Fields with the E(DATETIME) format are composed of eight digits of **DATE** in YYYYMMDD format and six digits of **TIME** in HHMMSS format. The combined rules of E(DATE) format and E(TIME) format apply.

Formats	Dec. 31, 2008 13:20:59 (sample)	Empty Value (default)	Jan 1,1 00:00:00(minimum)	Dec. 31, 9999 23:59:59 (maximum)
U.14	Z'20081231132059'	Z'00000000000000'	Z'00010101000000'	Z'99991231235959'
P,8	X'020081231132059F'	X'000000000000000F'	X'000010101000000F'	X'099991231235959F'
B,4 and F,4	F'20081231132059'	F'0'	F'10101000000'	F'99991231235959'

E(TIMESTAMP) Edit Mask

Fields with the E(TIMESTAMP) format are composed from the 14 digits of DATETIME and the six digits of microseconds.

Formats	Dec. 31, 2008 13:20:59 123456 (sample)	Empty Value (default)	Jan 1,1 00:00:00 000 000 (minimum)	Dec. 31, 9999 23:59:59 999 9 (maximum)
U.20	Z'20081231132059123456'	Z'00000000000000000000'	Z'00010101000000000000'	Z'99991231235959999999
P,11	X'020081231132059123456F'	X'000000000000000000000F'	X'000010101000000000000F'	X'099991231235959000000

E(NATDATE) Edit Mask

Fields with the E(NATDATE) format contain the number of days since January 2 of year 0. January 1, 0001 is equivalent to NATDATE(0) + 365.

The following information about dates in Natural should be considered:

- A RESET on a D field returns a blank or 0 (zero) if it is redefined as P7.
- The lowest date that can be displayed is D'1582-0101'. This is the year when the Gregorian calendar was first introduced.
- The maximum year that can be used in a Natural session is governed by the Natural profile parameter MAXYEAR. The default is 2699. Alternatively, it can be set to 9999.

In Adabas, we allows NATDATE values between 1582-01-01 and 9999-12-31.

The proleptic Gregorian calendar (produced by extending the Gregorian calendar backward to dates preceding its official introduction in 1582) is used as the basis for converting NATDATE to year, month, day. For complete information, read *Conversions Between Date-Time Representations*, elsewhere in this section.

Formats	Dec. 31, 2008 (sample)	Empty Value (default)	Jan 1,1 (minimum)	Dec. 31, 9999 (maximum)
P,4	X'0733771F'	X'0000000F' after RESET	Invalid	X'3652423F'
U,7	Z'0733771'	Z'0000000'	Invalid	Z'3652423'
B,3 and F,4	X'03B7F9'	X'00000000'	Invalid	X'37BB47'

Here are some sample dates:

Sample Dates	Value
0001-01-01 (outside range)	365
1582-01-01 (minimum)	577813
1970-01-01	719527
2699-12-31 (MAXYEAR default)	985789
9999-12-31 (MAXYEAR=9999)	3652423

E(NATTIME) Edit Mask

Fields with the E(NATTIME) format contain the number of days since January 2 of year 0 multiplied by 86400 * 10 plus the number of 0.1 seconds in a day.

A RESET on a T field returns January 2 of year 0. This is 365 days before the DATE epoch 0001-01-01.

The lowest date and time which can be set as a non-zero value is E'1582-01-01 00:00:00.0'. This is the year when the Gregorian calendar was first introduced.

In Natural, the profile parameter MAXYEAR specifies what the maximum year allowed in a Natural session. The default is 2599, but it can be set to 9999.

Adabas allows values between 1582-01-01 and 9999-12-31.

Formats	Dec. 31, 2008 13:20:59 (sample)	Empty Value (default)	Jan 1,1 (minimum)	Dec. 31, 9999 (maximum)
P,7	X'06339786 24591F'	X'00000000 00000F' after RESET	Invalid	X'31556943 35999F'
U,13	Z'6339786 24591'	Z'000000 0000000'	Invalid	Z'3155694 335999'
B,6 and F,8	X'939C121A4F'	X'000000000000'	Invalid	X'02DEBE 0A37FF'

Here are some sample dates:

Sample Dates	Value
0001-01-01 00:00:00.0 (outside range)	315360000
1582-01-01 00:00:00.0 (minimum)	499230432000
1970-01-01 00:00:00.0	621671328000
2699-12-31 23:59:59.9 (MAXYEAR default)	852037055999
9999-12-31 23:59:59.9 (MAXYEAR=9999)	3155694335999

E(UNIXTIME) Edit Mask

Fields with the E(UNIXTIME) format contain the number of seconds since January 1, 1970.

Normally UNIXTIME is UTC-based and conversion to local time is performed by the user application. When using it with local time values, Adabas will do no timezone adjustments or validity checking.

The maximum value of the 32-bit integer is 2147483647 or X'7FFFFFFF', which is equivalent to 2038-01-19 03:14:07.

Adabas will allow negative values if the underlying format is not B. Then, for a fixpoint field of length 4, the earliest date and time are 1901-12-13 20:45:52 (-2147483648 seconds or X'80000000').

Using 64-bit integers (format F, length 8) allows the full range of 0001-01-01 to 9999-12-31.

Using binary format B, 4, the range is from 0 (representing 1970-01-01 00:00:00) to a maximum value of X'FFFFFFFF' (representing 2106-02-06 06:28:15).

Formats	Dec. 31, 2008 13:20:59 (sample)	Empty Value (default)	Jan 1,1 (minimum)	Dec. 31, 9999 (maximum)
F4, B,4	1230729659	0	Invalid -- value truncation	Invalid -- value truncation
F,8	1230729659	0	-62135596800	253402300799
B,5	X'00495B71BB'	0	Invalid -- negative value	X'3AFFF3317F'
P,7	X'01230729659F'	X'00000000000F'	-62135596800	X'0253402300799F'
U,12	Z'1230729659'	Z'0000000000'	Z'--62135596800'	253402300799

E(XTIMESTAMP) Edit Mask

Fields with the E(XTIMESTAMP) format contain the number of microseconds since January 1, 1970.

Normally UNIXTIME is UTC-based and conversion to local time is performed by the user application. When using it with local time values, Adabas will do no timezone adjustments or validity checking.

Adabas will allow negative values if the underlying format is not B.

Formats	Dec. 31, 2008 13:20:59 (sample)	Empty Value (default)	Jan 1,1 (minimum)	Dec. 31, 9999 (maximum)
F,8	1230729659 123456	0	-62135596800 000000	253402300799 999999
B,8	X'045F57 94258700'	0	invalid	X'0384440C CC735FFF'
P,10	X'000123072965 9123456F'	X'000000000000 0000000F'	X'62135596800 000000D'	X'0253402300799 999999F'
U,18	Z'1230729659 123456'	Z'0'	Z'--62135596800 000000'	Z'253402300799 999999'

Conversions Between Date-Time Representations

The following table describes how non-zero, non-null date-time data is converted between date-time formats. Conversion rule letters are described at the end of the table. For information on the zero and null data conversions, read *Default and Null Value Handling*, elsewhere in this section.

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	X
DATE	Allowed. No date-time conversion is required for read or update commands.	n/a	Allowed. Time portion will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. Time and microseconds will be zero. Uses conversion rule F for read commands and rule T for update commands. .	Allowed. Conversion is required and uses rule C for both read and update commands. Date-time format limits are checked.	Allowed. Time values and tenths of seconds will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. Time will be zero. Uses conversion rule F for read commands and rule T for update commands.	Al Ti an mi wi Us co rul co an up co
TIME	n/a	Allowed. No date-time conversion is required for read or update commands.	n/a	n/a	n/a	n/a	n/a	

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAM
DATETIME	Allowed. Time will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The date will be omitted. Uses conversion rule X for read commands. Conversion is not allowed for update commands.	Allowed. No date-time conversion is required for read or update commands.	Allowed. Microseconds will be zeros. Uses conversion rule F for read commands and rule T for update commands .	Allowed. Time will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The tenths of a seconds will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. Conversion is required and uses rule C for both read and update commands.	Allowed. Microsecon will be zer Uses conversion rule F for re commands and rule T update commands
TIMESTAMP	Allowed. Time and microseconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The date will be omitted. Uses conversion rule X for read commands. Conversion is not allowed for update commands.	Allowed. Microseconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. No date-time conversion is required for read or update commands.	Allowed. Time and microseconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Values below the tenths of a second resolution will be zero. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Microseconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Conversion required a uses rule C both read a update commands
NATDATE	Allowed. Conversion is required and uses rule C for both read and update commands.	n/a	Allowed. The time will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. Time and microseconds will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. No date-time conversion is required for read or update commands.	Allowed. Time and tenths of seconds will be zero. Uses conversion rule F for read commands and rule T	Allowed. Time will be zero (rounded to the start of the day value). Uses conversion rule F for read commands and rule T for update commands.	Allowed. Time and microsecon will be zer Uses conversion rule F for re commands and rule T update commands

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	X
						for update commands.		
NATTIME	Allowed. Time and tenths of seconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The date will be omitted. Uses conversion rule X for read commands. Conversion is not allowed for update commands.	Allowed. The tenths of seconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Values below the tenths of a second resolution will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. The time and tenths of a second will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. No date-time conversion is required for read or update commands.	Allowed. The tenths of a second will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Al Va the sec res wi Us co rul co an up co
UNIXTIME	Allowed. Seconds in the day (time) will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The date will be omitted. Uses conversion rule X for read commands. Conversion is not allowed for update commands.	Allowed. Conversion is required and uses rule C for both read and update commands.	Allowed. Microseconds will be zero. Uses conversion rule F for read commands and rule T for update commands.	Allowed. Seconds in the day (time) will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Uses conversion rule F for read commands and rule T for update commands.	Allowed. No date-time conversion is required for read or update commands.	Al Mi wi Us co rul co an up co
XTIMESTAMP	Allowed. Microseconds in the day (time) will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. The date will be omitted. Uses conversion rule X for read commands. Conversion is not allowed for	Allowed. Microseconds in the day (time) will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Conversion is required and uses rule C for both read and update commands.	Allowed. Microseconds in the day (time) will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Allowed. Values below the tenths of a second resolution will be omitted. Uses conversion rule T for read	Allowed. Microseconds will be omitted. Uses conversion rule T for read commands and rule F for update commands.	Al da co rec rea up co

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAM
		update commands.				commands and rule F for update commands.		

Conversion rule letters have the following meanings:

Rule Letter	Meaning
C	The source and target value have the same precision.
F	Fill up the value with zeros; the target value has the higher precision. Date-time format conversion may be required depending on the date-time edit mask in use.
T	Truncate the value; the target value has the lower precision. Date-time format conversion may be required depending on the date-time edit mask in use.
X	Extract the time component from the value; the target value does not contain date information. Date-time format conversion may be required, depending on the date-time edit mask in use.

Conversions Between Date-Time Representations for Fields with the TZ option

The following table describes how non-zero, non-null date-time data is converted between date-time formats for fields with the TZ option. For information on the zero and null data conversions, read *Default and Null Value Handling*, elsewhere in this section.

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAMP
DATETIME	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	UTC times are not converted according to timezone for either read or update operations.	UTC times are not converted according to timezone for either read or update operations.
TIMESTAMP	Date-time conversions	Date-time conversions	Date-time conversions	Date-time conversions	Date-time conversions	Date-time conversions	UTC times are not	UTC times are not

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAMP
	from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	from UTC to local time and from local time to UTC are allowed for both read and update operations.	from UTC to local time and from local time to UTC are allowed for both read and update operations.	from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	from UTC to local time and from local time to UTC are allowed for both read and update operations.	converted according to timezone for either read or update operations.	converted according to timezone for either read or update operations.
NATTIME	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	UTC times are not converted according to timezone for either read or update operations.	UTC times are not converted according to timezone for either read or update operations.
UNIXTIME	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	Date-time conversions from UTC to local time are allowed in read operations. No date-time conversions are allowed in update operations.	Date-time conversions from UTC to local time and from local time to UTC are allowed for both read and update operations.	UTC times are not converted according to timezone for either read or update operations.	UTC times are not converted according to timezone for either read or update operations.
XTIMESTAMP	Date-time conversions from UTC to local time are allowed	Date-time conversions from UTC to local time are allowed	Date-time conversions from UTC to local time and from	Date-time conversions from UTC to local time and from	Date-time conversions from UTC to local time are allowed	Date-time conversions from UTC to local time and from local time to	UTC times are not converted according to timezone	UTC times are not converted according to timezone for either read

Input Format	Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAMP
	in read operations. No date-time conversions are allowed in update operations.	in read operations. No date-time conversions are allowed in update operations.	local time to UTC are allowed for both read and update operations.	local time to UTC are allowed for both read and update operations.	in read operations. No date-time conversions are allowed in update operations.	UTC are allowed for both read and update operations.	for either read or update operations.	or update operations.

Default and Null Value Handling

Depending on the numerical format of the field, format 0 is returned for empty values.

- For DATE, DATETIME, TIMESTAMP, NATDATE, and NATTIME edit masks, empty values are distinct from a valid date-time value.
- For TIME, UNIXTIME, and XTIMESTAMP edit masks, empty values are equal to the epoch of that format. In other words, for TIME format, the epoch is 00:00:00; for UNIXTIME or XTIMESTAMP formats, the epoch is January 1, 1970 00:00:00.

  With these formats, a proper value cannot be distinguished from a null value unless the NC option is specified for the field in the FDT.

For most date-time fields, a zero default value will be converted to zero in the target date-time format. However, be careful when processing zero default values using UNIXTIME or XTIMESTAMP formats. Conversion of a zero default value from UNIXTIME or XTIMESTAMP to other date-time formats (except TIME) will result in a value equivalent to the date 1970-01-01, rather than 0 (zero). This can lead to the filling up of empty fields. For example, consider a situation where a DATETIME field defined in the FDT is performing reads and updates using the UNIXTIME format, effectively not changing the zero value in the record buffer. An update would convert and store this value as the date 1970-01-01, which is not the default value 0 (zero) in DATETIME format. In addition, displaying these date-time values would appear as 1970-01-01 instead of 0 (zero).

The following table depicts the asymmetry of zero default value conversions between date-time formats.

Input Format	Converted Zero Output Format							
	DATE	TIME	DATETIME	TIMESTAMP	NATDATE	NATTIME	UNIXTIME	XTIMESTAMP
DATE	0	n/a	0	0	0	0	0	0
TIME	n/a	0	n/a	n/a	n/a	n/a	n/a	n/a
DATETIME	0	0	0	0	0	0	0	0
TIMESTAMP	0	0	0	0	0	0	0	0
NATDATE	0	n/a	0	0	0	0	0	0
NATTIME	0	0	0	0	0	0	0	0
UNIXTIME	1970-01-01	0	1970-01-01	1970-01-01	1970-01-01	1970-01-01	0	0
XTIMESTAMP	1970-01-01	0	1970-01-01	1970-01-01	1970-01-01	1970-01-01	0	0

Value Verification

The following value verification processing is performed on date-time field values:

- Unless the field value is zero, date-time field values in DATE, TIME, DATETIME, or TIMESTAMP formats are always checked for valid components. For example, they are checked to determine that any year values lie in the range 1-9999.
- Date-time fields in NATDATE, NATTIME, UNIXTIME, or XTIMESTAMP formats are checked to ensure they do not exceed the equivalent TIMESTAMP maximum value of "9999-12-31 23:59:59.999999".
- Date-time fields in NATDATE and NATTIME formats may be zero. Otherwise, they must greater than or equal to the value 1582-01-01.
- Date-time fields in UNIXTIME and XTIMESTAMP formats are checked to ensure they are not smaller than the TIMESTAMP minimum value of "0001-01-01 00:00:00.000000".
- Date-time field values must fit into the target fields; they are checked to ensure they do not exceed the target field lengths.

5 Expanded Files

- Defining Expanded Files 94
- Inserting a Component File 96
- Removing a Component File 96
- Deleting Expanded Files 97
- Inspecting an Expanded File 97
- Expanded Files and the Adabas Nucleus 97
- Expanded Files and Adabas Utilities 98

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 4294967294 records. An Adabas component file with 3-byte ISNs can contain a maximum of 16777215 records; a component file with 4-byte ISNs can contain 4294967294 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:	1-16000000
File 17, ISN range:	20000001-36000000
File 23, ISN range:	36000001-50000000

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***, elsewhere in this guide 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***, elsewhere in this guide 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 we recommend 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, ADADRU, ADARES, and ADASAV functions:

ADADBS DELETE Function

Deletes a complete expanded file only.

ADADRU EXPGROUP Function

If the EXPGROUP parameter is not specified, expanded files are processed without looking for expanded file associations (as component files).

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

The following utilities have functions that can check single component files:

- ADACDC, ADACMP, ADASEL, ADAULD
- ADADBS
- ADAINV
- ADALOD UPDATE Function
- ADAORD REORFILE / REORFASSO / REORDATA Functions

- ADASCR (Adabas Security) Functions

ADACDC, ADACMP, ADASEL, 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.

6 Large Object (LB) Files and Fields

- Getting Started with Large Object (LB) Fields 104
- LOB File Management 107
- Processing LOB Field Segments 107

Adabas 8 introduces large object fields (LB fields). These are fields that may contain much more data than the 253 bytes of normal alphanumeric fields or the 16381 bytes of LA fields. Fields containing large objects are defined with the new LB option. The theoretical maximum size of the value of an LB field is just short of 2 GB; practical usable sizes are smaller.

Adabas stores LB field values in a separate file, called a *LOB file*, that is tightly associated with the file containing the LB fields, called the *base file*.

Getting Started with Large Object (LB) Fields

To use LB fields, complete the following steps:

1 Define an FDT with one or more fields assigned the LB option. You can use the ADACMP COMPRESS function to create LB fields. For example:

```
ADACMP COMPRESS
ADACMP FNDEF='1,AA,8,A,DE'   Key field
ADACMP FNDEF=...
ADACMP FNDEF='1,AZ,250,A,NU'  Data field
ADACMP FNDEF='1,L1,0,A,LB,NV,NU,NB'  Binary LOB field
```

This defines field L1 as a large object field. The NV and NB options specify that this field is not subject to character conversion, nor to the compression of trailing blanks. The NU option specifies that this field is null-suppressed (fields with the NB option must also have either the NU or NC option defined). In effect, Adabas will not modify the provided field values in any way. For further information about the LB field option and all field options (including the NV and NB field options), read *Field Options*, in *Adabas Concepts Manual*.

If you provide input data for ADACMP to compress, the input can contain either short LB field values (up to 253 bytes) or large LB field values (greater than 253 bytes). In the uncompressed input record, at the place that corresponds to the LB field (in the example, L1), you may provide an empty LB value by specifying an inclusive length of X'00000004' that is not followed by LB data (since the LB value is empty). For further information about the structure of the COMPRESS input data, read *Input Data Requirements*, in *Adabas Utilities Manual*

Or:

You can also update an existing FDT using the ADADBS NEWFIELD function with an FNDEF specification defining an LB field. (Adabas Online System also provides equivalent functionality.) If you elect to do this, you do not need to define and load a new *base file* and you can skip the next step and proceed with **Step 3** of these instructions.

2 Use the ADALOD LOAD function to load the (possibly empty) base file with LB fields into the database.

You will load an empty LOB file separately in the next step, although the LOB and base files can be loaded in either order, or even in parallel. For example:

```
ADALOD LOAD FILE=11,NAME='BASE-FILE'
ADALOD      LOBFILE=12
ADALOD      MAXISN=100000,DSSIZE=5000B
ADALOD      ...
ADALOD      SORTSIZE=100,TEMPSIZE=100
```

The LOBFILE parameter specifies the file number of the LOB file associated with this base file. For complete information about the LOB-related ADALOD parameters, read *LOAD: Load a File*, in *Adabas Utilities Manual*.

Note: A base file-LOB file pair can also be established by just loading a *LOB file*, if a file with LB fields (*base file*) is already present.

Specify the //DDAUSBA output data set from the ADACMP COMPRESS step as //DDEBAND input data set for the ADALOD LOAD function.

3 Use the ADALOD LOAD function to load an empty *LOB file* and associate it with the *base file* you loaded in the previous step or that you updated in **Step 1**. (The LOB and base files can be loaded in either order, or even, in parallel.)

```
ADALOD LOAD FILE=12,LOB,NAME='LOB-FILE'
ADALOD      BASEFILE=11
ADALOD      MAXISN=500000,DSSIZE=500000B,NISIZE=4000B,UISIZE=40B
ADALOD      ISNREUSE=YES,DSREUSE=YES,INDEXCOMPRESSION=YES
ADALOD      ...
ADALOD      SORTSIZE=100,TEMPSIZE=100
```

The LOB file type indicates that ADALOD should load a LOB file with a predefined FDT. No //DDEBAND input data set need be supplied if you are loading an empty LOB file.

The BASEFILE parameter specifies the file number of the *base file* associated with this LOB file. For complete information about the LOB-related ADALOD parameters, read *LOAD: Load a File*, in *Adabas Utilities Manual*.

Note: A base file-LOB file pair can also be established by just loading a *LOB file*, if a file with LB fields (*base file*) is already present.

4 Run a database report using ADAREP and see how the base file and LOB file are shown in the report.

5 Reevaluate and adjust the Adabas nucleus (ADARUN) parameters NISNHQ, NH, LP, LDEUQ, LWP, NAB, and LU to make sure the nucleus will have sufficient resources for processing LB fields. For more information about these parameters, read *Adabas Initialization (ADARUN Statement)*, in *Adabas Operations Manual*. For information on how these parameters might need adjusting, read ***Getting Started with Large Object (LB) Fields***, elsewhere in this guide.

6 Write or adjust an application program to load data, including LB field values, into the base file. Behind the scenes, Adabas will store LB field values (except for very short ones) in the LOB file, but this is transparent to your application. Commands in application programs should always be directed against the base file; application programs need neither know nor care about the existence of a LOB file.

- One way to work with LB fields is using Natural 4.2. The LB field (for example, L1) of the LOB file is mapped to a dynamic variable in Natural. Otherwise, your application program can be written in much the normal way. Natural will automatically issue the Adabas calls in the format necessary to deal with large LB field values.
- Alternatively, your application program can issue direct calls against Adabas. For calls with *large* LB field values (more than 32 KB of data), you must use the ACBX direct call interface of Adabas 8. (Calls using the ACB direct call interface can be used if all specified LB field values fit into a single 32 KB buffer.) For more information, read *Specifying an ACBX Interface Direct Call*, in *Adabas Command Reference Guide*. For example, you could specify the following information in the ACBX direct call:

Direct Call Component	Specification

ACBX	`ACBXCMD = N1 (insert record)`
	`ACBXFNR = 11 (base file number)`
	`...`

First format/record buffer segment pair	`FB1 = 'AA,8,A,L1L,4,B,...,AZ,250,A.'` `RB1 = 'KEY-1  ' X'000186A0' ... 'Some arbitrary ↩ data...'`
Second format/record buffer segment pair	`FB2 = 'L1,*.'` `RB2 = X'100,000 bytes of binary data'`

The locations and lengths of the two buffer segment pairs must be given in four Adabas buffer descriptions (ABDs), which are not shown here. For more information, read *Adabas Buffer Descriptions (ABDs)*, in *Adabas Command Reference Guide*

The length indicator (L) for the LB field in the first format buffer segment (`L1L,4,B`) specifies that the length of the L1 field value proper is stored at the corresponding place in the first record buffer segment, which indicates an LB field value length of 100000 bytes (in hex, 186A0). For more information about the length indicator, read *Length Indicator (L)*, in the *Adabas Command Reference Guide*.

The asterisk (*) length notation in the second format buffer segment (`L1,*`) specifies that the LB field value proper (without any further length information) is stored at the corresponding location in the second record buffer segment. For more information about asterisk length notation, read *Asterisk (*) Length Notation* , in the *Adabas Command Reference Guide*.

LOB File Management

LOB files must be managed independently from their associated base files. For example, using ADADBS REFRESH to refresh the base file will not automatically refresh the LOB file; a separate, independent ADADBS REFRESH job for the LOB file must be run independently. Likewise, Adabas Online System (AOS) file functions must be applied separately to the LOB file from the base file.

Processing LOB Field Segments

LOB fields (LB-type fields) can be processed in segments (parts) using two different mechanisms:

- You can use a special *LOB segment notation* in the format buffer of an Adabas direct call to select part of a LOB field (a LOB segment) that should be processed by the call.
- You can read or write a LOB value in parts (LOB segments) from left to right using multiple A1 or L1/L4 calls.

This section describes each of these processing mechanisms.

- Format Buffer LOB Segment Notation Processing
- Multiple Call LOB Segment Processing

Format Buffer LOB Segment Notation Processing

You can select an entire LOB field or part of a LOB field (a LOB segment) in the format buffer of a direct call using the following syntax:

```
field-name [index-or-range] [(bytenum,LOBlength[,LOBlength2])]
```

The last part of this syntax `(bytenum,LOBlength[,LOBlength2])` is used to identify the segment of a LOB field that should be processed. This part of the syntax is therefore called *LOB segment notation*. For complete information on this syntax, read *Selecting LOB Values or LOB Value Segments* , in the *Adabas Command Reference Guide*.

Using LOB segments you can:

- Read only a part of a LOB value;
- Delete or replace the rightmost part of a LOB value;
- Replace a LOB segment with a new segment of equal length.

This section describes the rules for each of these uses of a LOB segment.

- Reading a LOB Segment
- Maintaining the Rightmost Part of a LOB
- Replacing a LOB Segment

Reading a LOB Segment

To read a segment of a LOB value, the `(bytenum,LOBlength)` section of LOB segment notation is used. The following rules apply:

1. The number of bytes specified by `LOBlength` at the corresponding position in the record buffer are filled with data from the database. If `LOBlength` is zero, no data is put into the record buffer.
2. The LOB value segment to be read starts at the position in the LOB value specified by `bytenum` and is `LOBlength` bytes long. If an asterisk (*) is specified for `bytenum`, the segment to be read starts at the current position in the LOB value:
   - For an L1 or L4 command with L specified in the Command Option 2 field (ACBCOP2 or ACBXCOP2) of the Adabas control block, the current position is derived from the ISN Lower Limit field (ACBISL or ACBXISL) of the Adabas control block, which specifies the number of bytes preceding the LOB value segment to be read. If the current position points past the end of the value, response code 3 (ADARSP003) is issued, indicating the end of the LOB read sequence. An ISN Lower Limit value of zero corresponds to byte number 1. Response code 3 (ADARSP003) is given if the ISN Lower Limit value is greater than or equal to the length of the LOB value.

     For an L1 or L4 command with the L option, the new current position after the read of the segment is returned in the ISN Lower Limit field. It equals the old current position plus the `LOBlength`. The new current position may point past the end of the LOB value.
   - For a read or search command without the L option, the current position is defined to be the leftmost byte of the LOB value (byte number 1). No response code 3 (ADARSP003) is given if the LOB value is empty (i.e., has length zero).
3. If the ending position points past the end of the LOB value, the LOB value is considered to be padded with a sufficient number of blanks to make the padded value include the entire segment requested.
4. The record buffer space associated with the format element is filled with the requested segment of the LOB value, including any necessary padding with blanks.

Maintaining the Rightmost Part of a LOB

To delete the rightmost part of a LOB value, or to replace the rightmost part of a LOB value with new data, the LOB segment notation `(bytenum,LOBlength)` is used. The following rules apply:

1. The data to be deleted from the LOB value starts at the insertion point identified by `bytenum` and extends to the end of the LOB value. New data, if any, is inserted at the same insertion point. If an asterisk (*) is specified for `bytenum`, the insertion point is the current position in the LOB value:
   - For an L1 or L4 command with L specified in the Command Option 2 field (ACBCOP2 or ACBXCOP2) of the Adabas control block, the current position is derived from the ISN Lower Limit field (ACBISL or ACBXISL) field of the Adabas control block, which specifies the number of bytes preceding the LOB value segment to be replaced. An ISN Lower Limit value of zero corresponds to byte number 1. For an A1 command with the L option, the new current position after the replacement of the segment is returned in the ISN Lower Limit field. It equals the old current position plus `LOBlength`. If no trailing blanks were removed after the segment was replaced, the new current position points to the end of the LOB value. If trailing blanks were removed, the new current position points past the end of the LOB value stored in the database.
   - For a store or update command without the L option, the current position is defined to be the leftmost byte of the LOB value (byte number 1).
2. Any existing data in the LOB value at and to the right of the insertion point is deleted or replaced. If the insertion point is at the end of the LOB value, no data is deleted and the new data is appended to the LOB value.
3. If the insertion point lies past the end of the LOB value, the value is padded with a sufficient number of blanks to extend to the insertion point.
4. The number of bytes specified by `LOBlength` from the corresponding position in the record buffer, constituting the segment to be written, are stored at the insertion point. If `LOBlength` is zero, no bytes are stored.
5. Unless the LOB field has the NB option (no blank compression), any blanks at the end of the resulting LOB value are removed. If no data or only blanks were inserted at the end of the value, the compression operation removes any trailing blanks to the left of the insertion point.

Note: For a LOB field with the NB option (no blank compression), the length of the LOB value after the write operation always equals the position of the insertion point (minus 1) plus `LOBlength`. If the insertion point lies past the end of the LOB value and no data is to be inserted (`LOBlength` is zero), the LOB value is padded with blanks up to the insertion point. (For a LOB field without the NB-option, this padding does not make any difference, as the inserted blanks are removed by the compression operation.)

Replacing a LOB Segment

To replace a segment of a LOB value with a new segment of the same length, the extended LOB segment notation `(bytenum,LOBlength,LOBlength2)` is used (where the value of `LOBlength2` must equal the value of `LOBlength`). The following rules apply:

1. The LOB value segment to be replaced starts at the insertion point identified by `bytenum` and is `LOBlength` bytes long. If an asterisk (*) is specified for byte-number, the insertion point is the current position in the LOB value:
   - For an A1 command with L specified in the Command Option 2 field (ACBCOP2 or ACBX-COP2) of the Adabas control block, the current position is derived from the ISN Lower Limit field (ACBISL or ACBXISL) field of the Adabas control block, which specifies the number of bytes preceding the LOB value segment to be replaced. An ISN Lower Limit value of zero corresponds to byte number 1. For an A1 command with the L-option, the new current position after the replacement of the segment is returned in the ISN Lower Limit field. It equals the old current position plus `LOBlength`. The new current position may point past the end of the LOB value stored in the database.
   - For a store or update command without the L option, the current position is defined to be the leftmost byte of the LOB value (byte number 1).
2. If the ending position points past the end of the LOB value, the LOB value is considered to be padded with a sufficient number of blanks to make the padded value include the entire segment.
3. The number of bytes specified by `LOBlength` from the corresponding position in the record buffer, constituting the segment to be written, are stored at the insertion point, replacing any existing data of the same length stored at and past this position. If `LOBlength` is zero, no data is stored.
4. Unless the LOB field has the NB option (no blank compression), any blanks at the end of the resulting LOB value are removed. If only blanks were inserted at the end of the value, the compression operation removes any trailing blanks to the left of the insertion point.

   Note: For a LOB field with the NB option (no blank compression), if the insertion point lies past the end of the LOB value and no data is to be inserted (length is zero), the LOB value is padded with blanks up to the insertion point.

Multiple Call LOB Segment Processing

The command option L in the Command Option 2 field of L1/L4 and A1 commands allows you to read and write a LOB value from left to right with multiple calls using a constant format. This section describes this processing.

- Reading a LOB Value using Multiple Calls
- Writing a LOB Value using Multiple Calls
- L Option Rules
- Current (Asterisk) Positioning in LOB Values

Reading a LOB Value using Multiple Calls

To read a LOB value from left to right with multiple calls using a constant format, an application should use a sequence of L1 (or L4) commands on the record containing the LOB value. The format buffer should use the LOB segment notation `(bytenum,LOBlength)` to position each read call. It can reference the current position in the LOB field using an asterisk (*) in the `bytenum` specification (current positioning notation). The L option should be specified in the Command Option 2 field of the Adabas control block to request tracking of the current position in the LOB value using the ISN Lower Limit field of the Adabas control block. Here is a recommended command sequence for sequential LOB reading:

1. The application uses any read-type command to locate and read the target record, excluding the LOB value. It may put the record in shared or exclusive hold status to prevent parallel updates to the current record while the following read sequence on the LOB field is in progress.

   Note: We strongly recommend that you protect the integrity of a LOB value while it is being read piecemeal in multiple calls. This may be done using application rules that prevent possible concurrent LOB updates (for example, the application might ensure that LOB values are never changed and become visible to readers only once they have been fully stored), or by putting the record containing the LOB value in shared or exclusive hold status. If you are reading a LOB value piecemeal and you update the LOB value during the read sequence, it is your responsibility, too, to ensure that the value being read is consistent in terms of the application logic.

2. The application issues an L1/L4 command to read the first segment of the LOB value. The format buffer references the LOB field using an asterisk in `bytenum` (this is current positioning notation). The L option is set in the Command Option 2 field of the control block to indicate that the current position in the LOB value should be tracked using the ISN Lower Limit field of the control block. Initially, the application sets the ISN Lower Limit field to zero, which is interpreted as position 1 of the LOB field. Adabas increments the ISN Lower Limit field by the number of bytes read (specified in `LOBlength`).
3. Until response code 3 (ADARSP003) is issued to indicate that the end of the LOB value has been reached, the application issues further L1/L4 commands with the same format buffer as in step 2. The L option is set in the Command Option 2 field of each call to indicate that LOB field

reading should continue at the current position in the LOB value, which is identified by the ISN Lower Limit field. When the application is only reading the LOB value from left to right, it should not modify the ISN Lower Limit field; Adabas will increment the ISN Lower Limit setting each time by the number of bytes read.

4. When done with reading the LOB value, the application may continue to work on the current record (for example, update it) or go on to the next record.

Writing a LOB Value using Multiple Calls

To write a LOB value from left to right with multiple calls using a constant format, an application should use a sequence of A1 commands on the record containing the LOB value. The format buffer should use the LOB segment notation `(bytenum,LOBlength)` to position each write call. It can reference the current position in the LOB field using an asterisk (*) in the `bytenum` specification (current positioning notation). The L option should be specified in the Command Option 2 field of the Adabas control block to request tracking of the current position in the LOB value using the ISN Lower Limit field of the Adabas control block. Here is a recommended command sequence for sequential LOB writing:

1. The application uses an N1/N2 or A1 command to store or update the target record, initially with an empty LOB value. The mandatory exclusive hold status of the record prevents parallel updates to the current record while the following write sequence on the LOB field is in progress.

   Note: The exclusive hold status of the record does not prevent parallel reads of the current record, including the LOB value, unless those reads request shared or exclusive hold status for the record, too. If the commands used to write or update the LOB value are spread over more than one transaction, it is the user's responsibility to ensure that either the LOB value cannot be updated between transactions or the next transaction will detect and adapt to a change of the value.

2. The application issues an A1 command to write the first segment of the LOB value. The format buffer references the LOB field using an asterisk in `bytenum` (this is current positioning notation). The L option is set in the Command Option 2 field of the control block to indicate that the current position in the LOB value should be tracked using the ISN Lower Limit field of the control block. Initially, the application sets the ISN Lower Limit field to zero, which is interpreted as position 1 of the LOB field. Adabas increments the ISN Lower Limit field by the number of bytes written (specified in `LOBlength`).
3. Until the entire LOB value has been written, the application issues further A1 commands with the same format buffer as in step 2. The L option is set in the Command Option 2 field of each call to indicate that LOB field writing should continue at the current position in the LOB value (at the end of the value), which is indicated by the ISN Lower Limit field. When the application is only writing the LOB value from left to right, it should not modify the ISN Lower Limit field; Adabas will increment the ISN Lower Limit setting each time by the number of bytes written.
4. When done with writing the LOB value, the application may continue to work on the current record (for example, read or update it again) or go on to the next record.

L Option Rules

The following rules apply to the use of the L option in Command Option 2 of the Adabas control block in conjunction with the use of current (asterisk) positioning in the format buffer:

1. The Command Option 2 field can only be set to L in the control blocks for L1, L4 and A1 commands. If L is specified, the format buffer must contain exactly one format element for a LOB field segment using current (asterisk) positioning.
2. When the L option is specified, the ISN Lower Limit field in the Adabas control block must be a number in the range from 0 to 2147483647 (i.e., 2^{31-1}). This number gives the current position in the LOB value, specified as the number of bytes preceding the target segment. Adabas increments the ISN Lower Limit field by the `LOBlength` specified in the format buffer. (The returned ISL value lies in the range from 0 to 4294967294.)
3. When the L option is specified in a read command (L1 or L4), response code 3 (ADARSP003) with a new subcode is given if the current position in the LOB value points past the end of the value (if the value in the ISN Lower Limit field is greater than or equal to the length of the LOB value).
4. The L option cannot be specified with the I or N options of an L1/L4 command, nor with the V option of an A1 command. (The V option is defined in Adabas on open systems only and is related to ADAM files.) All of these options are specified in the Command Option 2 field and they also imply the processing of a new record, whereas the L option is used for the reading or writing of (different parts of) the same record with multiple calls.
5. The L option cannot be specified with the M or P options. Instead of reading multiple consecutive LOB value segments in a single command using multifetch or prefetch, the application should rather read a single larger segment.

Current (Asterisk) Positioning in LOB Values

When an L1/L4 and A1 command sequence is performed with the L option (using the current (asterisk) position notation in the format buffer element), the current position in the LOB value being read or written must be tracked. Multiple LOB read/write sequences may be in progress in parallel, with a different position being current in each LOB value.

The ISN Lower Limit (ISL) field in the Adabas Control Block is used to keep track of the current position in the LOB value being read or written. Prior to the call, the application program sets the ISN Lower Limit field to the number of bytes preceding the LOB value segment to be read, replaced, or deleted. After the call, Adabas increments ISN Lower Limit by the length specified for the LOB field segment in the format buffer.

To read an existing (or write a new) LOB value from left to right using a format element with current (asterisk) positioning, your application should initially set the ISL field to zero for the first L1/L4 or A1 command of the read (or write) sequence. For the subsequent calls, it should leave the ISN Lower Limit field unchanged or it must reproduce the value previously returned by Adabas.

The ISN Lower Limit field is used to determine the current position in a LOB value only if the L option is specified, too. For a call without the L option, the current position is defined to be byte number 1 of the value and the ISL field is neither used nor updated in the course of the operation on the LOB value segment. Only L1, L4, and A1 commands can be issued with the L option.

7 Defining an Adabas Database

- Step 1 : Estimate the Size of the Database 116
- Step 2 : Allocate Space 134
- Step 3 : Format the Space 136
- Step 4 : Define Database Parameters 136

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 steps described in this chapter.

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
- Other Components
- General Space Requirements
- General Procedure for Estimating Space
- Estimation Formulas
- Normal Index (NI)
- Upper Index (UI)
- Address Converter (AC)
- Data Storage
- How Adabas Allocates Work Space
- Work Part 1: Data Protection Information
- Work Part 2: Intermediate Search Results
- Work Part 3: ISN Lists from Search Commands
- Work Part 4: Data Related to Distributed Transaction Processing
- Sort

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 and 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 data set, DDWORKR4 is required. DDWORKR4 is a container data set 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 data set 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
- Logs

Sort and Temp Areas

The Adabas utilities ADACHK, ADAINV, and ADALOD require two additional data sets, 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 data sets can be allocated during the job and then released, or permanent data sets 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) data sets can be used (multiple data sets 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) data sets can be used (multiple data sets 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 data set is limited to 16777215 (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 equaling 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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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 * 20000 + 50 * (3 + 2)
                    = 60000 + 250
                    = 60250 bytes
```

```
NIRBLOCKS  = 60250 / (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 * 20000 + 20000 * (10 + 2)
                    = 80000 + 240000
                    = 320000 bytes
```

```
NIRBLOCKS  = 320000 / (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 * 10000 + 300 * (4 + 2)
                    = 210000 + 1800
                    = 211800 bytes
```

```
NIRBLOCKS  = 211800 / (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 * 10000 + 10 * (5 + 2)
                    = 80000 + 70
                    = 80070 bytes
```

```
NIRBLOCKS  = 80070 / (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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.

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  = (2000000 * 3) / 2004
                     = 6000000 / 2004
                     = 2994.01
                     = 2995 blocks
```

Example 2:

MAXISN=2000000
RABNSIZE=4

```
ACBLOCKS  = (2000000 * 4) / 2004
                     = 8000000 / 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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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 block size for DATA = 4820
Data Storage block padding factor = 5 (%)

```
DATASTORAGE  = 1000000 / ((4820 * (1 - 0.05)) / 50)
                              = 1000000 / (4579 / 50)
                              = 1000000 / 91
                              = 10989.01
                              =  10990 blocks
```

How Adabas Allocates Work Space

When you allocate the Work data sets, 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:

Notes:

1. You can use the DRES operator command to monitor your system's use of Work parts 1, 2, and 3. For more information, read *DRES Command*, in the *Adabas Operations Manual*.
2. 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 data set. DDWORKR4 is a container data set used for the same purpose as Work part 4, but it can be used in parallel by all members in a cluster. The DDWORKR4 data set 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 data set.

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 data set 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 compressed 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

<table>
<tr><td>WORK2</td><td colspan="2">is the Work part 2 space requirement, in blocks.</td></tr>
<tr><td rowspan="5">RECORDS</td><td colspan="2">is the number of records in the file with the most records. This number equals

TOPISN - MINISN + 1

where:</td></tr>
<tr><td>TOPISN</td><td>is the highest ISN currently used in the file.</td></tr>
<tr><td>MINISN</td><td>is the lowest ISN used in the file.</td></tr>
<tr><td colspan="2">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.</td></tr>
<tr><td>BLKSIZE</td><td colspan="2">is the block size of the device where the Work data set is stored. To review a list of block sizes by device type and component, refer to the sections entitled Device And File Considerations in the Adabas installation documentation for z/OS.</td></tr>
</table>

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 data set is stored on a 3380 device.

```
WORK2 = 22 + 2 * ((4 * 500000) / (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 data set. DDWORKR4 is a container data set used for the same purpose as Work part 4, but it can be used in parallel by all members in a cluster. The DDWORKR4 data set 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 required size of Work part 4 depends on the applications running against the database and on the system load. If you have Adabas Transaction Manager Version 7.4 or earlier installed, a safe size of LP/4 is a good value to start with. If you have Adabas Transaction Manager 7.5 or later installed, a minimum size of 8 blocks must be specified, with the maximum number of blocks being the size of the DDWORK4 data set divided by 8.

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 can aid in *estimating* the sort data space needed for sorting all values of a single descriptor. While they are complicated, they assess all factors needed to estimate the sort usage. Adabas Online System (AOS) also provides estimates for the sort size and may be simpler for you if AOS (the non-demo version) is available to you. The calculations used in AOS are even more complex but are done internally to simplify the process.

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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.

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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for z/OS.
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 data sets 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 data set. However, you can allocate the Associator and Data Storage on up to 99 separate data sets each; the data sets can be allocated on the same or different device types. Note that your actual real maximum number of physical extents may be less than 99 because the maximum number you can define is derived from the block size of the first Associator data set (DDASSOR1).
2. To minimize contention and distribute I/O activity more evenly across hardware channels, place the ASSO, DATA, WORK, PLOG, and RLOG data sets on different physical volumes. If only two volumes are available, place ASSO on one volume and DATA and WORK data sets on the second.

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

The RLOG data set 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) of the format utility (ADAFRM). See the Adabas Utilities documentation for specific information and job examples.

   - Examples
   - Performance Note

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 GmbH does not recommend using hardware compression (IDRC) for protection log files. The ADARES utility BACKOUT function will run at least twice as long under z/OS when processing compressed data.

Step 3 : Format the Space

Before loading the first file into the database, use the ADAFRM utility to format the ASSO, DATA, and WORK data sets. 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 data sets and delete them after executing the utility, or allocate and format permanent data sets for repeated use.

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

Format the CLOG, PLOG, and RLOG data sets 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 data sets must be defined with ADADEF DEFINE parameters. Note that defining the sizes to Adabas is different from allocating the space; the data sets must be allocated and formatted before you can define them to Adabas. The sizes of the other data sets 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 data set (CLOG, PLOG, or RLOG) is limited to 16777215 (x'FFFFFF') blocks/RABNs.

8 Database Space Management

- Adabas Physical Extents 138
- Adabas Device Type 138
- Relative Adabas Block Number (RABN) 139
- Adabas Logical Extents 140
- Adabas Space Allocation and Deallocation 141
- Using the Database Status Report to Control Space Use 150
- Potential Space Use Problems and Recommended Actions 150
- Receiving Associator and Data Storage Space Usage Notifications 152
- Increasing Associator and Data Storage Space 153
- Secondary Space Considerations when Defining and Extending Adabas Data Sets 156

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.

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 about 99 Adabas physical extents. However, your actual real maximum could be less because the extent descriptions of all Associator, Data Storage, and Data Storage Space Table (DSST) extents must fit into the general control blocks (GCBs). For example, on a standard 3390 device type, there could be more than 75 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there are less for another).

Adabas Device Type

The block sizes of each Adabas database container are determined by the Adabas device type. Device types are set during container initialization. The device type is typically defined using the ADARUN DEVICE= parameter, but may be overridden during database setup, for example via the DEVICE= subparameter of ASSOFRM/DATAFRM/WORKFRM/etc when formatting the container using the ADAFRM utility, and via the ASSODEV, DATADEV and WORKDEV parameters when defining the database using the ADADEF utility.

The default Adabas device type (ADARUN DEVICE= parameter) is 3390. Many new disk devices have appeared since the original 3390 disk drives and 3990 control units were introduced, but on z/OS, the 3390 has become an architectural standard in recent years, and new disk devices now emulate the 3390. This standard was introduced by IBM to ensure compatibility. When migrating to new disks, users need only to check with IBM that the new disks emulate 3390.

Adabas provides a set of device types which are compatible with 3390. These are 3390, 8390, 8391, 8392 and 8393. There is also a set of device types which are compatible with the older 3380 disk devices. These are becoming obsolete, but will remain available to Adabas as long as z/OS continues to support them.

For more details on the characteristics of the Adabas device types, see *Installation for z/OS > Device and File Considerations*.

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.

When space is allocated for Adabas, it must be designated as Associator (ASSO), Data Storage (DATA), the Work area (WORK), logging area (PLOG, CLOG, RLOG, ALOG), sort area, or temp area. A block allocated to ASSO using device 3390 contains 2544 bytes, but a block allocated to DATA using device 3390 contains 5064 bytes. Block sizes are predefined for each device type and Adabas component. To review a list of block sizes by device type and component, see *Installation for z/OS > Device and File Considerations*.

Adabas identifies and addresses each physical block within a database component (Associator, Data Storage, Work) 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 within each database component, starting with "1". If multiple physical extents are used, the RABN assignment continues across the physical extents.

The first track of the first physical extent of the Associator, Data Storage and Work components is not used. The first track of the second and each subsequent physical extent as well as all extents of TEMP, SORT, CLOG, and PLOG are used.

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 16777215.

- If RABNSIZE=4 (block number is 31 bits or four bytes), the maximum number of RABNs is 2147483646.

The number of Adabas blocks that can be stored on a given physical unit (track/cylinder/volume) of external storage is different for each database component and for each device type.

Using the information provided in *Supported Device Types*, in *Adabas z/OS Installation Guide*, the number of blocks in the container may be calculated as shown in the following examples. In addition, the RABN ranges stored on each volume can easily be determined using the Adabas Online System report function or the ADAREP utility.

Example 1: Associator database component, device type 3390 comprising 880 cylinders

```
number of ASSO blocks = blocks/track * tracks/cylinder * number of cylinders
= 18 * 15 * 880 = 237,600 Associator blocks
```

Eighteen (18) blocks must be subtracted for the first track of the first Associator physical extent; therefore, the Associator volume contains 237,582 blocks.

Example 2: Data Storage database component, device type 8393 comprising 750 cylinders

```
number of DATA blocks = blocks/track * tracks/cylinder * number of cylinders
= 2 * 15 * 750 = 22,500 Data Storage blocks
```

Two (2) blocks must be subtracted for the first track of the first Data Storage physical extent; therefore, the Data Storage volume contains 22,498 blocks.

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 two address converters may be allocated for any file, one for the file itself and one for any spanned records in the file. If spanned records are not used, only one address converter is allocated.

The total number of logical file extents that can be defined is limited only in that the extent information of all address converters, Data Storage, normal index, and upper index extents for the file must fit into the file control block (FCB). (The extent information is stored in a variable section of the FCB.) For example, on a standard 3390 device type, a file could have more than 40 extents of each type (or there could be more of one type if there are less for another).

When the Adabas nucleus starts up, the FCBs are checked. If there is insufficient FCB space for four further extents, the Adabas nucleus prints messages 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. Normal users trying to access the file will then get response code 17 (ADARSP017) or 48 (ADARSP048).

The "Contents of the Database" section of an ADAREP utility report flags files that may have insufficient FCB space for ten further extents. In the "File Information" section of the report, ADAREP prints a conservative estimate for how many further file extents can possibly be built for each file.

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
- Adabas Nucleus Space Allocation Rules
- Space Allocation with the ADADBS Utility
- Space Allocation with the ADAINV Utility
- Space Allocation with the ADALOD Utility
- Space Allocation by the ADAORD Utility
- Space Allocation by ADASAV (RESTORE FILES Function)

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.

Adabas Nucleus Space Allocation Rules

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

This section describes the rules used for the allocation.

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. For 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 INCREASE 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* data set to the Associator or Data Storage. The new data set 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 99 data sets each. However, your actual real maximum will be less because the maximum number of physical extents that you can define is derived from the block size of the first Associator data set (DDASSOR1). For example, on a standard 3390 device type, there could be more than 75 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there are less for another).

The INCREASE function results in the physical extension of an *existing* data set. 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 data set 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, 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 to allow additional 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 zap-able 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 or DSDEV parameters may be used to force the allocation to a specific RABN 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 zap-able 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 99 extents are possible. However, your actual real maximum will be less because the maximum number of physical extents that you can define is derived from the block size of the first Associator data set (DDASSOR1). If space is not available, ADALOD ends with an error message.

The maximum number of logical file extents that you can now define is derived from the block size of the first Associator data set (DDASSOR1). The extent information is stored in a variable section of the FCB. New extents can be added now until the used FCB size reaches the block size of the Associator data set.

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 data set, 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 data set 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 adddelete 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 99 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 number of Associator physical extents (about 99) 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 number of Data Storage physical extents (about 99) 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.

Receiving Associator and Data Storage Space Usage Notifications

You can set up your Adabas database so that you receive space usage messages as Associator (ASSO) and Data Storage (DATA) space use exceeds or falls below set thresholds. With this information, you can adjust the use or size of your database as needed.

Two new ADARUN parameters are provided to specify ASSO and DATA space threshold levels. When free space becomes critically low, space usage messages can be useful triggers for operators to determine the cause of the limited space. If the cause is a rogue application, you can cancel the rogue application. If the cause is insufficient database space allocations, you can provide additional space while the database remains active (as described in this document).

- ASSOSPACEWARN can be used to specify the space usage thresholds and the increments above the thresholds at which warning messages should be triggered for Associator space.
- DATASPACEWARN can be used to specify the space usage thresholds and the increments above the thresholds at which warning messages should be triggered for Data Storage space.

When space use exceeds the specified thresholds, space usage messages are issued to the console. Thereafter, when space use increases or decreases by the specified increments, additional space usage messages are issued to the console. When space use drops below the lowest threshold specification, a space usage message is issued when it first falls below the threshold, but incremental space usage messages are no longer issued unless space use increases above the lowest threshold again.

In addition, two new operator commands, `ASSOSPACEWARN` and `DATASPACEWARN`, can be used to alter the comparable ADARUN parameter settings while the nucleus is active.

Increasing Associator and Data Storage Space

On z/OS systems, you can automatically and dynamically increase the size of your Associator (ASSO) or Data Storage (DATA) space. This makes it possible to increase the size of the database without incurring a database outage, which can be critical if your database needs to run 24x7 (24 hours a day, 7 days a week). You can do this in several ways:

1. You can request that Adabas automatically increase the size of your ASSO or DATA space using a series of AUTOINC* ADARUN parameters in the nucleus startup job.
2. You can dynamically alter the AUTOINC* ADARUN parameters using a series of AUTOINC* operator commands.
3. You can dynamically increase the size of the last existing ASSO or DATA data set for the database by extending the data set and formatting the appended space, and then allocating it to the database.
4. You can dynamically add a new ASSO or DATA data set to the database by allocating the data set to the operating system, formatting the data set, and then allocating it to the database.

The last three methods can be accomplished without restarting the Adabas nucleus. However, if you change the nucleus startup job to include the AUTOINC* ADARUN parameters, the nucleus must be stopped and restarted to pick up the changes. Once restarted, however, the automatic increases occur without restarting the nucleus again.

Note: You can also request that space usage messages be issued as Associator (ASSO) and Data Storage (DATA) space use exceeds or falls below set thresholds. This request is specified by the `ASSOSPACEWARN` and `DATASPACEWARN` ADARUN parameters. The settings for these parameters can then be altered while the nucleus is active using the `ASSOSPACEWARN` and `DATASPACEWARN` operator commands. For more information about these ADARUN parameters and operator commands, read *Adabas Initialization (ADARUN Statement)* and *Operator Commands*, in the *Adabas Operations Manual*.

The remainder of this section provides procedures for each of these methods:

- Automatically Increasing Associator or Data Storage Data Set Space
- Dynamically Adjusting the Automatic Increase Settings
- Dynamically Increasing Existing Associator or Data Storage Data Set Space

- Dynamically Adding New Associator or Data Storage Data Sets

Automatically Increasing Associator or Data Storage Data Set Space

You can request that Adabas automatically increase the size of your Associator (ASSO) or Data Storage (DATA) space when usage reaches a set threshold. This makes it possible to increase the size of the database without incurring a database outage, which can be critical if your database needs to run 24x7 (24 hours a day, 7 days a week). This support is implemented using the following ADARUN parameters:

- The `AUTOINCASSOSIZE` and `AUTOINCDATASIZE` parameters can be used to specify the size, in cylinders or blocks, by which the highest-numbered (last-defined) Associator (ASSO) and Data Storage (DATA) data sets (respectively) will be increased by an automatic database increase process.
- The `AUTOINCASSOTHRESHOLD` and `AUTOINCDATATHRESHOLD` parameters can be used to specify the Associator and Data Storage (respectively) space thresholds at which Adabas automatically initiates a database increase process. These thresholds are expressed as the percentage of used Associator or Data Storage space to total Associator or Data Storage space available in the database.

  For `AUTOINCASSOTHRESHOLD` and `AUTOINCDATATHRESHOLD` parameters to be checked, it does require `MODE=MULTI` to be specified. If `MODE=SINGLE` is in effect, then the threshold parameters are ignored, due to the amount of time that may be required to perform the increase.
- The `AUTOINCASSOTOTAL` and `AUTOINCDATATOTAL` parameters can be used to specify the maximum total size, in cylinders or blocks, of the entire Associator or Data Storage space (respectively) of the database that is not to be exceeded by automatic database increase processes.

For complete information about ADARUN parameters, read *Adabas Initialization (ADARUN Statement)*, in the *Adabas Operations Manual*.

Dynamically Adjusting the Automatic Increase Settings

The AUTOINC* ADARUN parameters can be used to request that Adabas automatically increase the size of your Associator (ASSO) or Data Storage (DATA) space when usage reaches a set threshold. These ADARUN parameter settings can also be altered dynamically by the following operator commands, which can be entered on the console as well as using the ADADBS OPERCOM utility function and using Adabas Online System's Modify Parameters screen.

- The `AUTOINCASSOSIZE` and `AUTOINCDATASIZE` commands can be used to specify the size, in cylinders or blocks, by which the highest-numbered (last-defined) Associator (ASSO) and Data Storage (DATA) data sets (respectively) will be increased by an automatic database increase process.
- The `AUTOINCASSOTHRESHOLD` and `AUTOINCDATATHRESHOLD` parameters can be used to specify the Associator and Data Storage (respectively) space thresholds at which Adabas automatically initiates a database increase process. These thresholds are expressed as the percentage of used

Associator or Data Storage space to total Associator or Data Storage space available in the database.

For `AUTOINCASSOTHRESHOLD` and `AUTOINCDATATHRESHOLD` parameters to be checked, it does require `MODE=MULTI` to be specified. If `MODE=SINGLE` is in effect, then the threshold parameters are ignored, due to the amount of time that may be required to perform the increase.

- The `AUTOINCASSOTOTAL` and `AUTOINCDATATOTAL` commands can be used to specify the maximum total size, in cylinders or blocks, of the entire Associator or Data Storage space (respectively) of the database that is not to be exceeded by automatic database increase processes.

For complete information about Adabas operator commands, read *Operator Commands*, in the *Adabas Operations Manual*. For complete information about the ADADBS OPERCOM utility, read *OPERCOM: Issue Adabas Operator Commands*, in the *Adabas Utilities Manual*.

Dynamically Increasing Existing Associator or Data Storage Data Set Space

To increase the last existing data set of the Associator or Data Storage of the database without shutting down the Adabas nucleus, use the ADADBS ONLINCREASE utility function to dynamically increase the size of the last existing ASSO or DATA data set for the database. Running this utility function will trigger Adabas to extend the data set, format the appended space, and allocate the space to the database.

For more information, read *ONLINCREASE: Allocating Appended ASSO or DATA Space Dynamically*, in the *Adabas Utilities Manual*. Pay special attention to the restrictions listed for this utility function.

Dynamically Adding New Associator or Data Storage Data Sets

To add a data set to the available Associator or Data Storage space without shutting down your Adabas database, complete the following steps:

1. Use the ADAFRM ASSOFRM or ADAFRM DATAFRM utility functions to create and format the data set.

   Only the data set specified by the function name and the NUMBER parameter is accessed and formatted. The FROMRABN parameter cannot be specified when formatting a new data set. For complete information about the ADAFRM utility, read *ADAFRM Utility: Format Adabas Database Components*, in the *Adabas Utilities Manual*.
2. Add appropriate JCL statements for the new data set to the nucleus startup job and to any Adabas utility procedures you have defined.
3. Use the ADADBS ONLADD utility function to dynamically allocate the added ASSO or DATA data set to the database, without restarting the nucleus. Specify the JCL statement for the new data set in that utility job.

Note: The ADADBS ONLADD function must be performed in a different job than the preceding ADAFRM function.

For complete information about the ADADBS ONLADD utility function, read *ONLADD: Allocating New ASSO or DATA Space Dynamically*, in the *Adabas Utilities Manual*. Pay special attention to the restrictions listed for this utility function.

Secondary Space Considerations when Defining and Extending Adabas Data Sets

In this section, "extent" refers to the space allocated to a data set by z/OS on a disk volume.

When defining and formatting Adabas data sets via ADAFRM, the JCL SPACE parameter allows for a secondary space allocation size to be defined.

The Adabas utilities used subsequently when extending data sets typically do not take account of the original secondary space allocation set up when the file was defined. Instead, Adabas attempts to satisfy the extend request with a single secondary extent. This is done in order to minimize the number of extents getting allocated, as the allocation is limited by z/OS to 16 extents per volume for a single data set. It also allows support for extending data sets which were originally defined with a secondary space allocation of zero.

Note that multiple extents might still get used when processing a single request to extend a data set. For example, a request to add another 4000 cylinders to a data set might result in two disk extents, one of 3000 cylinders and one of 1000 cylinders. This is done by z/OS, according to its own internal space allocation algorithms.

When extending data sets, there must be enough space on a single volume to satisfy the request. Therefore, if a large amount of additional space is requested, the secondary space size might get limited by Adabas. For a basic format data set, the size is limited to 4369 cylinders. For a large format data set, the size is limited to 65520 cylinders. When Adabas limits the extent size, the increase request will still be honored, but will result in multiple extents being allocated.

There is an exception to the handling of secondary extents described above. The ADACHK utility can dynamically extend SORT data sets. If the ADACHK utility detects that the SORT data set is fully formatted with the expected block size, Adabas will use the original secondary space allocation. This is done in order to prevent a pre-formatted SORT data set from occupying more disk space than originally planned.

9 Database Monitoring and Tuning

- Monitoring Resource Use 158
- Reporting on Resource Use 158
- Monitoring Database Controls 158
- Performance Management, Statistics, and Tuning 159
- Adabas Session Statistics 160
- Command Logging 168

This chapter describes the data base administrator's tasks in the area of monitoring and tuning.

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 benefitting 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 include input/output (I/O statistics, command statistics, buffer/queue usage statistics, highest usage session statistics, and various additional session statistics.

This section covers the following topics:

- Input/Output Statistics
- Command Statistics
- Buffer and Queue Statistics
- High Usage Session Statistics
- Additional Session Statistics

Input/Output Statistics

The following I/O statistics are provided:

- I/O Counts (Including Initialization)
- WORK1 and PLOG write I/O counts (if ADARUN LFIOP>0)
- Distribution of ASSO/DATA I/Os by VOLSER Number (Excluding Initialization)

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).

WORK1 and PLOG write I/O counts (if ADARUN LFIOP>0)

Statistic	Count
PLOG protection blocks	1603
PLOG different blocks	1086
PLOG protection I/Os	1015
WORK1 protection blocks	1596
WORK1 different blocks	1023
WORK1 protection I/Os	995

Write I/O statistics for the database protection areas, PLOG and WORK1 (Work part 1), are provided when the asynchronous buffer flush is enabled (ADARUN LFIOP>0). The total count of protection blocks written, the count of different protection blocks written, and the count of protection write I/Os are provided. The total count includes blocks that were written more than once (with increasing amounts of data). The count of different blocks includes each block only once. The I/O count is lower than the total block count when some I/Os write out multiple blocks in one operation.

For example, if WORK RABN 101 is half full and written out (to make a transaction persistent during an ET command), then filled to completion and subsequently written out together with RABN 102, statistics would be calculated as follows:

- the count of WORK1 protection blocks submitted for write-out would be 3 (101 twice and 102 once);
- the count of different WORK1 protection blocks submitted would be 2 (101 and 102); and
- the count of protection I/Os would be 2 (first 101 alone, then 101 and 102).

These counts can assist with the tuning of database update processing, particularly for update-intensive applications, as well as indicate the efficiency of the settings for ADARUN parameters NWORK1BUFFERS and NPLOGBUFFERS. A large gap between the total count of protection blocks written and the count of different blocks written indicates that many blocks were written out before they were full, usually in order to complete ET commands. In addition to these counts, the high watermarks for NWORK1BUFFERS and NPLOGBUFFERS, if at or close to 100%, may indicate potential for reducing the protection write I/Os by increasing these parameters.

Distribution of ASSO/DATA I/Os by VOLSER Number (Excluding Initialization)

VOLSER	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 12687 calls in five threads.

- Distribution of Commands by Source
- Distribution of Commands by Thread
- Distribution of Commands by File
- Distribution of Commands by Type

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	0
LOCAL	0
INTERNAL	0
OPERATOR	12686

Distribution of Commands by Thread

The following table shows the thread activity for the session:

Thread	Number
1	7328
2	2728
3	1240
4	814
5	541
TOTAL:	12651

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	4247
1	8404
TOTAL:	12651

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	4198
ET	4191
L1/4	4242
OP	56
TOTAL:	12687

The command type UC indicates a privileged call issued by Adabas utilities.

Note: The command type REST indicates commands such as C2 and RS.

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[1]	%
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
PLOG-IO	NPLOGB=1	1	100
SC	LCP= 10000	0	0
TBI	LI = 10000	0	0
TBS	LQ = 10000	0	0
UQ	NU = 20	4880	86
UQF[2]	NU = 20		
Wrkl-IO	NWORK1=1	1	100
WORK	LWP = 14000	70464	50
Work Part 1	LP=1500	59	3
Work Part 2	LWKP2=372	0	0
Work Part 3	*No parameter*[3]=818	0	0
XID	XID = 0	0	0

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.

Notes:

1. The High-Water Mark values represent the used bytes of the pool. The pool sizes depend on the setting of the appropriate ADARUN parameter. The ADARUN parameter either specifies the size of the pool in bytes (for example, LDEUQP, LFP, LCP, and LI) or the number of elements (for example, NAB, NC, NH, and NU). When the ADARUN parameter specifies the number of elements, the calculation of the pool size determined by multiplying the number of elements by the specific element size. The element size is described in the specific description of the ADARUN parameter.
2. The UQF is the user queue extension that holds the file list. The size of its pool is computed using the UQ pool size.

3. The Work Part 3 value represents the remaining size of the Work data set. It is calculated by subtracting the LP and LWKP2 settings from the total Work data set size (`Work Part 3 = WorkDSsize - LP - LWKP2`).

High Usage Session Statistics

The following high usage session statistics are produced for a nucleus session:

```
There were     nnn  users    participating

Most calls (       nnn)    initiated by: User   'userid'
                                         Job    jobname
                                         SECUID secuid

Most I/O-s (       nnn)    initiated by: User   userid
                                         Job    jobname
                                         SECUID secuid

Most thread time (  hh:mm:ss)   used by: User   'userid'
                                         Job    jobname
                                         SECUID secuid
```

The following substitutions occur in these statistics:

Substitution	Description
hh:mm:ss	The highest thread time used by a user during the session.
jobname	The name of the job that initiated the maximum number of calls or I/Os during the session or the highest thread time during the session.
nnn	The maximum number of calls or I/Os performed by a user during the session or the number of users participating in the session.
secuid	If known, the security system IDs of the users who initiated the maximum number of calls or I/Os during the session or the highest thread time during the session.
userid	The user IDs (last eight bytes of the communication IDs) of the users who initiated the maximum number of calls or I/Os during the session or the highest thread time during the session. User IDs are printed in text format when they are printable; otherwise, they appear in hexadecimal format.

Note: The high usage session statistics include only users who ended their session with a CL command. Users whose sessions were terminated abnormally (for example, through a timeout) are not included.

Additional Session Statistics

The following additional session statistics are produced for a nucleus session:

nnn Formats had to be translated

nnn Formats had to be overwritten

nnn Autorestarts were done

nnn Throw-backs due to ISN problem

nnn Throw-backs due to space problem

nnn Buffer-flushes were done

These are described in more detail in this section.

- Formats Translated/Overwritten
- Autorestarts
- Command Throwbacks
- Buffer Flushes

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.

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.

10 Error Handling and Message Buffering

- User Exit Failures 170
- Recovery or Plug-In (PIN) Routines 171
- PIN Routine User Exit 181

The error handling and message buffering facility helps implement 24-by-7 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 GmbH.

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.

User Exit Failures

User exits and hyperdescriptor exits 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*, it 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 *not critical* 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 formatting is provided to aid in error analysis:

Interrupt 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 log. Note that interrupt 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.

Interrupt Code	Exception Type	The processor . . .
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 add-on products 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
PINUES

PINUES is 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 add-on product Adabas Fastpath. It is established automatically when the Adabas Fastpath server component initializes at nucleus startup (ADARUN FAST-PATH=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 data set 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 add-on product 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 data set 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 (ADARSP009), 65 (ADARSP065), 72 (ADARSP072), 88 (ADARSP088), 97 (ADARSP097), 99 (ADARSP099), 148 (ADARSP148), or 151 (ADARSP151) 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 add-on product 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 data set 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 data set 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 (ADARSP075) or 77 (ADARSP077) 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. PINRSP is also loaded automatically at startup if the ADARUN SMGT parameter is set to "YES", similarly to ADAMXY.

Note: Only response codes set by Adabas can be logged. A response code such as 22 (ADARSP022 - invalid command code), which is set by the Adabas SVC before it reaches Adabas, is not logged.

If PINRSP is loaded 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 (ADARSP177):
   - basic response code logging
   - the index structure from the thread
   - active CQEs
   - buffer pool headers
3. Response codes such as 40 (ADARSP040) where additional IUB areas may be pertinent:
   - basic response code logging
   - IUBs
   - active CQEs
4. Response codes such as 255 (ADARSP255) where additional attached buffer information may be necessary:
   - basic response code logging
   - active CQEs
   - attached buffer information

5. Response codes such as 72 (ADARSP072) 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 (ADARSP201), 202 (ADARSP202), or 203 (ADARSP203). This may be useful in situations where a new application receives enough security errors to fill the SYSLOG. Although Software GmbH does not recommend this action, you may temporarily modify ADASMXIT to suppress N77 messages and activate PINRSP with response codes 201 (ADARSP201), 202 (ADARSP202) and 203 (ADARSP203) 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 add-on product 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 data set 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 reactivated, 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 (ADARSP017), 48 (ADARSP048), and 55 (ADARSP055). 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 (ADARSP017) and 48 (ADARSP048) 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 (ADARSP055) 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 ADAN02 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 z/OS system link list. If you are running either Adabas Parallel Services or Adabas Cluster Services in z/OS environments, 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  OD
MAXARPIN DC  F'6'       Maximum of 6 files can be excluded from autorestart
BADRSPS  DC  XL1'48'    Response code 72 (ADARSP072) 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

When 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 (ADARSP201)/202 (ADARSP202)/203 (ADARSP203) 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 (ADARSP009).

The entry X'0105020311' indicates that response code 9 (ADARSP009), subcodes 2, 3, and 16 are logged. Response code 9 (ADARSP009) is logged a maximum of 5 times.

11 Universal Encoding Support (UES)

- Wide-Character Encodings 186
- Wide-Character Data Support 188

Note: UES support requires that you use a version 7 or above Adabas SVC or router.

The Universal Encoding Support (UES) is a database option that enables Adabas to:

- perform data conversions;
- handle wide-character encoding;
- 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 encoding is 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.

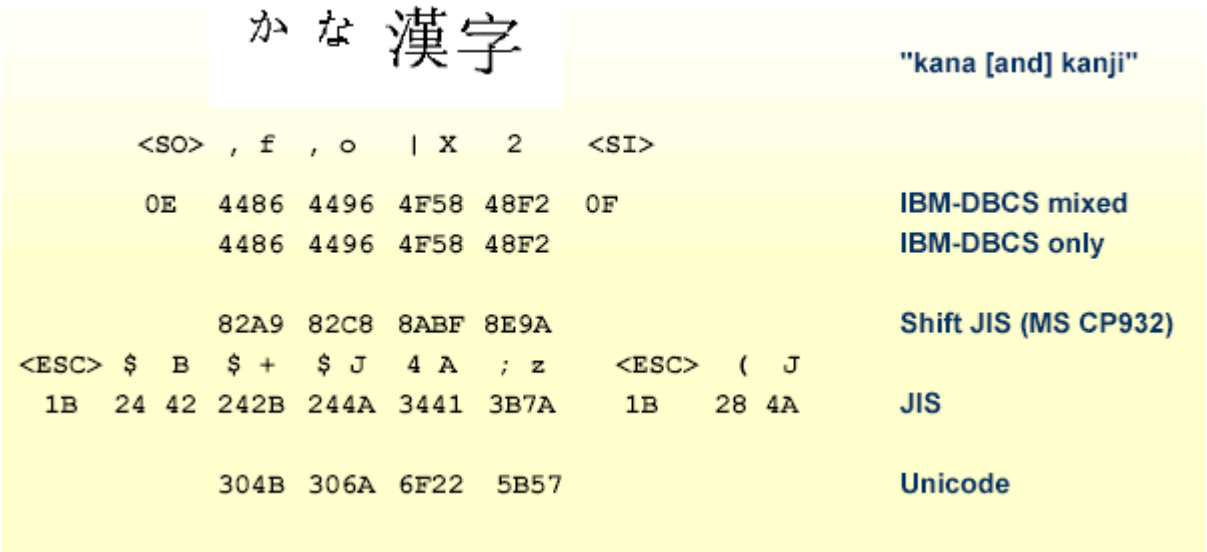
The default encoding for Wide format is Unicode for both storage and user. This default can be changed on user and storage level to the encoding appropriate for the intended usage.

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.

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 ICS; and
- native Unicode (the user encoding), the standard Java text encoding, can be directly stored and retrieved.

The disadvantages are as follows:

- 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

12 Multiple Platform Support

- Encodings 192
- ADACOX Conversion Exit 193
- Conversion of High Value in Value Buffer 193
- Data Translation Restrictions 194
- Platform Considerations 195

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 z/OS mainframe database through the TCP/IP protocol from web-based applications or from PC-based applications such as Software GmbH'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.

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 local 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 Technology Solutions, 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

A conversion exit, ADACOX, is available for your use. ADACOX supports *context-sensitive* conversion between Windows-1256 and EBCDIC 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.

This problem is solved 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 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 subdescriptors.

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.

13 Adabas SMF Records

- Record Structure 198
- Record Size Limits 198
- Record Subtypes 199
- Statistical Recording 200
- Record Sections 200
- ASMFREC Mapping Macro 217
- SMF User Exit 218
- IBM Type 89 SMF Records 219

On z/OS, Adabas supports the recording of database activity via SMF records when ADARUN parameter SMF=YES is specified.

SMF recording is supported on z/OS only.

All Adabas SMF records have a common structure, with some sections appearing in all records and others generated according to specific events and parameters specified through ADARUN or operator commands. The ASMFREC macro provides mapping DSECTs for all parts of the SMF record.

Record Structure

Adabas follows the modern convention for SMF record formats. A single record has:

- A standard IBM-type header
- A self-defining section that describes a variable number of detail sections
- A product ID detail section
- User-selected detail sections

Each detail section is described by an eight-byte entry in the self-defining section containing three fields. An entry is also called a *triplet*.

- A 4-byte offset from the beginning of the record to the detail section
- A 2-byte count of the number of instances of the detail section
- A 2-byte length of each detail section instance. If there are no detail section instances of a given type, the triplet is all zeros.

Field `ASNumD` in the **product ID section** specifies the number of triplets in the self-defining section.

Record Size Limits

SMF records are z/OS V-format records with a system-imposed maximum length of 32,756 bytes. Most Adabas SMF records fit within this limit for most reasonable types of ADARUN nucleus specifications. However, detail sections such as **File Activity** could potentially have several thousand detail section instances.

If the entire set of instances will not fit in the space remaining in the record, Adabas will include only as many as there are room for and write the record. The SMF record is reset by clearing the triplets for all detail sections except the product ID section and then adding as many of the remaining instances as will fit, repeating until all detail sections are processed. Field `ASSegNo` in the

product ID section will start at 1 and be incremented with each additional record, and field `ASSegL` will be set to zero for the last (or only) record for an interval or event.

If any detail section is so large that even one instance would cause the record size limit to be exceeded after resetting the SMF record, that detail type is deactivated.

Record Subtypes

The header section field `ASSTy` identifies a record subtype.

- Subtype 1 (ASStI) - Adabas Nucleus Initialization
- Subtype 2 (ASStT) - Adabas Nucleus Termination
- Subtype 3 (ASStI) - Adabas Interval Statistics
- Subtype 4 (ASStP) - Adabas Parameter Change

Subtype 1 (ASStI) - Adabas Nucleus Initialization

A record of this subtype is generated during nucleus initialization. In addition to the header, self-defining and product ID sections, it contains ADARUN parameter and user sections if these have been selected by SMFDETAIL ADARUN parameter or operator commands.

Subtype 2 (ASStT) - Adabas Nucleus Termination

A record of this subtype is generated during nucleus termination. In addition to the header, self-defining and product ID sections, it contains all detail sections specified by the SMFDETAIL ADARUN parameter or operator commands except for the ADARUN parameter section. Statistics in Adabas detail sections reflect totals for the entire nucleus session.

Subtype 3 (ASStI) - Adabas Interval Statistics

If interval recording has been specified by the SMFINTERVAL ADARUN parameter or operator command, a record of this subtype is generated at the expiration of each interval. In addition to the header, self-defining and product ID sections, it contains all detail sections specified by the SMFDETAIL ADARUN parameter or operator commands except for the ADARUN parameter section. Statistics in Adabas detail sections reflect activity since the previous interval ended, except where noted. This is also called a delta value.

Subtype 4 (ASStP) - Adabas Parameter Change

If the Adabas ADARUN parameter detail section has been specified by the SMFDETAIL ADARUN parameter or operator commands, a record of this subtype is generated whenever an ADARUN parameter value is changed after nucleus initialization. In addition to the header, self-defining, product ID and ADARUN parameter sections, it may also contain a user section if that has been selected by SMFDETAIL ADARUN parameter or operator commands.

Statistical Recording

The nucleus accumulates usage statistics on the resources it uses to accomplish its tasks. These statistics may be recorded at user or system-defined intervals (see ADARUN parameter SMFINTERVAL) and at termination.

Interval recording (Adabas SMF record subtype 3) provides the usage since the last interval ended for each detail section. Adabas SMF record intervals may be synchronized with one of the system-level intervals specified by PARMLIB member SMFPRM*xx* entries. This allows straightforward analysis of the usage by allowing direct comparison with other record interval data. For example, you can compare the Adabas interval record with RMF data for the same interval to better understand system performance.

Statistics at termination (Adabas SMF record subtype 2) will have cumulative statistics that reflect activity for the entire nucleus session in each specified detail section.

Record Sections

Every Adabas SMF record contains header, self-defining and product ID sections. You can select additional detail sections through the SMFDETAIL ADARUN parameter or operator commands. Each section is mapped by a DSECT generated by the ASMFREC mapping macro.

The following table summarizes the Adabas SMF record sections:

Detail Section Description	ASMFREC Macro or ADARUN Parameter Specification	Self-Defining Section Triplet Label Base	ASMFREC DSECT Name Produced by the ASMFREC Macro
Header and self-defining section	---	---	**ASBase**
Adabas command activity	CMD	ASTCmd	**ASCmd**
Adabas global cache activity by block type[1]	CSHB	ASTChB	**ASChB**
Adabas global cache activity by Adabas file number[1]	CSHF	ASTChF	**ASChF**

Detail Section Description	ASMFREC Macro or ADARUN Parameter Specification	Self-Defining Section Triplet Label Base	ASMFREC DSECT Name Produced by the ASMFREC Macro
Adabas global cache activity [1]	CSHG	ASTChG	**ASChG**
Adabas Parallel Services cache activity[2]	CSHP	ASTChP	**ASChP**
Adabas file activity	FILE	ASTFile	**ASFile**
Adabas global lock activity[1]	LOCK	ASTLok	**ASLok**
Adabas internucleus messaging control block activity	MSGB	ASTMsgB	**ASMsgB**
Adabas internucleus messaging counts	MSGC	ASTMsgC	**ASMsgC**
Adabas internucleus messaging service time histogram	MSGH	ASTMsgH	**ASMsgH**
ADARUN parameter values	PARM	ASTParm	**ASParm**
I/O by DD name	IODD	ASTIODD	**ASIODD**
Product ID[3]	ID	ASTPID	**ASPID**
Session Statistics	SESS	ASTSess	**ASSess**
Storage pool	STG	ASTStg	**ASStg**
Thread activity	THRD	ASTThrd	**ASThrd**
zIIP statistics	ZIIP	ASTzIIP	**ASzIIP**
User-defined	USER	ASTUsr	user-defined

1. The detail section is available only in cluster environments when either Adabas Cluster Services or Adabas Parallel Services are installed.
2. The detail section is available only in cluster environments when Adabas Parallel Services is installed.
3. The product ID section is always included in every SMF record. It may not be specified in the SMFDETAIL ADARUN parameter or in operator commands.

This section describes the different detail record sections.

Note: The DSECTs provided in the following sections may not be the most current. To see the most current versions of the DSECTs, generate them using the **ASMFREC macro**.

- Header Section
- Self-Defining Section
- Product ID Section: ID
- Adabas Command Activity Section: CMD
- Adabas File Activity Section: FILE
- Adabas Global Cache Activity by Block Type Section: CSHB
- Adabas Global Cache Activity by Adabas File Number Section: CSHF
- Adabas Global Cache Activity Section: CSHG
- Adabas Global Lock Activity Section: LOCK

- Adabas Internucleus Messaging Control Block Activity Section: MSGB
- Adabas Internucleus Messaging Counts Section: MSGC
- Adabas Internucleus Messaging Service Time Histogram Section: MSGH
- Adabas Parallel Services Cache Activity Section: CSHP
- ADARUN Parameter Value Section: PARM
- I/O by DD Name Section: IODD
- Session Statistics Section: SESS
- Storage Pool Section: STG
- Thread Activity Section: THRD
- zIIP Statistics Section: ZIIP

Header Section

IBM has defined a standard format for the initial part of all SMF records in *z/OS MVS System Management Facilities (SMF), IBM document SA22-7630*. This section begins every Adabas SMF record.

```
ASBase   DSect ,                   Base segment
*
*                       Standard SMF Header
*
ASRDW    DS    0Bl4                Record descriptor word
ASLen    DS     Bl2                 Record length
ASSeg    DS     Bl2                 Segment descriptor
ASFlg    DS    Bl.8                System indicator flags
ASFStV   Equ    x'40'               Subtypes are valid
ASFV4    Equ    x'10'               MVS/SP V4 and above
ASFV3    Equ    x'08'               MVS/SP V3 and above
ASFV2    Equ    x'04'               MVS/SP V2 and above
ASFVS2   Equ    x'02'               VS2
ASRTy    DS    Bl1                 Adabas record type
ASTme    DS    Bl4                 Time since midnight when record was +
                                       moved into SMF buffer in 1/100 sec
ASDte    DS    Pl4                 Date when record was moved into SMF +
                                        buffer as 0cyydddF
ASSID    DS    Cl4                 System identifier (SMFPRMxx SID)
ASSSI    DS    Cl4                 Subsystem identifier
ASSty    DS    Bl2                 Subtype
ASStI    Equ     1                  Adabas initialization
ASStT    Equ     2                  Adabas termination
ASStS    Equ     3                  Interval statistics
ASStP    Equ     4                  Parameter change
ASStA    Equ     9                  Ad hoc record
*
ASBaseL  Equ   *-ASBase            Length of standard header
```

Self-Defining Section

The self-defining section follows immediately after the header section. It is part of the header section DSECT.

Each detail section triplet is identified by a base label as shown in the **table at the beginning of this section**. The base label begins with the prefix specified in the ASMFREC invocation followed by the letter T (for triplet), and then followed by a mnemonic detail section identifier. The base label with suffix O is the offset, with suffix L is the length, and with suffix N is the number of instances.

Here is an example of some triplets.

```
*
*                       Self-Defining Section
*
ASSDS    DS    0B                  Self-defining section
*                                  Map of typical section triplet
ASSDS0   DS     B14                 Offset to section from start of    +
                                       record
ASSDSL   DS     B12                 Length of section
ASSDSN   DS     B12                 Number of section(s)
         Org    ASSDS
*
ASTID    DS    0B18                ID Section (always present)
ASTID0   DS     B14                 Offset to ID section from start    +
                                       of record
ASTIDL   DS     B12                 Length of ID section
ASTIDN   DS     B12                 Number of ID section(s)
*
ASTUser  DS    0B18                User-Defined Section
ASTUser0 DS     B14                 Offset to User-Defined section from+
                                       start of record
ASTUserL DS     B12                 Length of User-Defined section
ASTUserN DS     B12                 Number of User-Defined section(s)
*
ASTParm  DS    0B18                ADARUN Parameter Section
ASTParm0 DS     B14                 Offset to detail section from start+
                                       of record
ASTParmL DS     B12                 Length of each detail section
ASTParmN DS     B12                 Number of detail section(s)
*
 . . .
*
ASSDSLn  Equ   *-ASSDS             Length of self-defining section
ASSDSNT  Equ   ASSDSLn/8           Number of triplets
```

Product ID Section: ID

The product ID section is always present in every Adabas SMF record with one instance. It describes the nucleus generating the SMF record and provides information about the record's contents.

The 2-byte version code consists of a major version and a minor version. A change, such as adding a new triplet or extending a detail section, will increment the minor version. All existing programs should continue to operate as no existing displacements have changed. A more disruptive change will increment the major version and require existing programs to (at least) be reassembled.

```
ASPID    DSect ,                    Product ID Detail Section          +
                                      (always present in SMF record)
ASSMFV   DS    0Bl2                 SMF record version
ASSMFVM  DS     Bl1                  SMF record major version
ASSMFVN  DS     Bl1                  SMF record minor version
ASSMFVC  Equ   ASSMFV14             Current version: 1.4
ASSMFV11 Equ     x'0101'             Version 1.1 - Initial release
ASSMFV12 Equ     x'0102'             Version 1.2 - Added ALS/ASM
ASSMFV13 Equ     x'0103'             Version 1.3 - ADARUN parms V8.3
ASSMFV14 Equ     x'0104'             Version 1.4 - ADARUN parms V8.4
*                                      in coll. seq., new section SESS
ASSegNo  DS    Bl1                  Record segment number
ASSegL   DS    Bl1                  Last segment when = 0
ASNumD   DS    Bl2                  Number of detail type triplets
ASPNm    DS    Cl8                  Product name (ADABAS)
ASVRSC   DS    Cl8                  Product ver/rlse/SM/cum: vvrrsscc
ASSysN   DS    Cl8                  System name
ASSypN   DS    Cl8                  Sysplex name
ASVMN    DS    Cl8                  Virtual machine name
ASJbN    DS    Cl8                  Job name
ASStN    DS    Cl16                 ProcStep/Step name
ASJNm    DS    Cl8                  JES job identifier
ASPgm    DS    Cl8                  Program name
ASGrp    DS    Cl8                  Cluster messaging group name
ASST     DS    Bl8                  Nucleus start time in STCK format
ASIST    DS    Bl8                  Interval start time in STCK format
ASIET    DS    Bl8                  Interval end time in STCK format
ASDBID   DS    Bl4                  Database ID
ASNucX   DS    Bl2                  External nucleus ID
ASNucI   DS    Bl1                  Internal nucleus ID
ASSVC    DS    Bl1                  Adabas SVC number
ASASID   DS    Bl2                  Address space ID
ASASIDI  DS    Bl4                  Reusable address space ID instance
ASComp   DS    Bl2                  Completion code                    +
                                      x'0ccc' System ABEND code ccc     +
                                      x'8ccc' User ABEND code ccc
ASARC    DS    Bl4                  ABEND reason code
*
ASPIDL   Equ    *-ASPID
```

Adabas Command Activity Section: CMD

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Adabas command activity data is derived from data presented at nucleus shutdown.

There is one instance for each command group: A1/4, BT, CL, ET, E1/4, L1/4, L2/5, L3/6, L9, LF, N1/2, OP, UC, RC, RE, REST, S1/4, S2, S5, S8, S9, YA, YB, YF, YP, YCAL, V1, V2, V3, V4, U0, U1, U2 and U3. There are 34 possible instances but this is subject to change in future releases.

```
ASCmd    DSect ,                    Adabas Command Activity
ASCmdNm  DS     Cl4                 Command name
ASCmdCt  DS     Bl8                 Number of times this command type   +
                                      was executed
ASCmdTm  DS     Bl8                 Sum of this command type durations  +
                                      in microseconds
*
ASCmdL   Equ   *-ASCmd
```

Adabas File Activity Section: FILE

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Adabas file activity data is derived from data presented at nucleus shutdown or in response to a DFILUSE operator command. There is one instance for each file possible in the database as specified by ADADEF MAXFILES up to the highest file number with a non-zero use count. The file number is implied by the sequence number of the instance, starting with zero, which reflects commands such as OP that are not associated with a specific file.

```
ASFile   DSect ,                    Adabas File Activity
ASFileCt DS     Bl8                 Number of commands executed against +
                                      this file
*
ASFileL  Equ   *-ASFile
```

Adabas Global Cache Activity by Block Type Section: CSHB

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Global cache statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DX-CACHE operator command. There is one detail section instance for each type of block. Users should examine the block type and not rely on any observed order of the instances. The following block types are reported:

- AC: Address Converter
- DS: Data Storage
- DSST: Data Storage Space Table
- FCB: File Control Block

- NI: Normal Index
- UI: Upper Index
- WORK: Protection Data
- OTHR: Any other block type

```
ASChB     DSect ,                  Global Cache Activity by Block
ASCBCN    DS    Bl2                Cache Number
ASCBRsv1  DS    Bl2                Unused
ASCBBT    DS    Cl4                Block type
ASCBRT    DS    Bl8                Reads - Total
ASCBRCS   DS    Bl8                Reads - Completed synchronous
ASCBRCA   DS    Bl8                Reads - Completed asynchronous
ASCBRIC   DS    Bl8                Reads - Data in cache
ASCBRNI   DS    Bl8                Reads - Data not in cache
ASCBRFS   DS    Bl8                Reads - Failed - Structure
ASCBRO    DS    Bl8                Reads - For cast-out
ASCBROS   DS    Bl8                Reads - For cast-out synchronous
ASCBROA   DS    Bl8                Reads - For cast-out asynchronous
ASCBWT    DS    Bl8                Writes - Total
ASCBWCS   DS    Bl8                Writes - Completed synchronous
ASCBWCA   DS    Bl8                Writes - Completed asynchronous
ASCBWDR   DS    Bl8                Writes - Data written
ASCBWNR   DS    Bl8                Writes - Data not written
ASCBWSF   DS    Bl8                Writes - Structure full
ASCBVI    DS    Bl8                Validates issued
ASCBVF    DS    Bl8                Validates failed
ASCBBD    DS    Bl8                Block deletes issued
ASCBDR    DS    Bl8                Deletes reissued due to timeout
ASCBUR    DS    Bl8                Number of times updates redone
*
ASChBL    Equ  *-ASChB
```

Adabas Global Cache Activity by Adabas File Number Section: CSHF

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Global cache statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DX-CACHE operator command. There is potentially one instance for each file possible in the database as specified by ADADEF MAXFILES. The size of this detail section precludes the ability to generate it for every possible file, so there is one detail section instance for each file that has non-zero usage. Users should examine the file number and not rely on any observed order of the instances.

```
ASChF     DSect ,                  Global Cache Activity by File
ASCFCn    DS    Bl2                Cache Number
ASCFRsv1  DS    Bl2                Unused
ASCFNum   DS    Bl4                File number
ASCFRT    DS    Bl8                Reads - Total
ASCFRCS   DS    Bl8                Reads - Completed synchronous
ASCFRCA   DS    Bl8                Reads - Completed a synchronous
ASCFRIC   DS    Bl8                Reads - Data in cache
ASCFRNI   DS    Bl8                Reads - Data not in cache
ASCFRFS   DS    Bl8                Reads - Failed - Structure
ASCFRO    DS    Bl8                Reads - For cast-out
ASCFROS   DS    Bl8                Reads - For cast-out synchronous
ASCFROA   DS    Bl8                Reads - For cast-out asynchronous
ASCFWT    DS    Bl8                Writes - Total
ASCFWCS   DS    Bl8                Writes - Completed synchronous
ASCFWCA   DS    Bl8                Writes - Completed a synchronous
ASCFWDR   DS    Bl8                Writes - Data written
ASCFWNR   DS    Bl8                Writes - Data not written
ASCFWSF   DS    Bl8                Writes - Structure full
ASCFVI    DS    Bl8                Validates issued
ASCFVF    DS    Bl8                Validates failed
ASCFBD    DS    Bl8                Block deletes issued
ASCFDR    DS    Bl8                Deletes reissued due to timeout
ASCFUR    DS    Bl8                Number of times updates redone
*
ASChFL    Equ   *-ASChF
```

Adabas Global Cache Activity Section: CSHG

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Global cache statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DX-CACHE operator command. This detail section appears with one instance.

```
ASChG     DSect ,                  Global Cache Activity Section
ASCGCN    DS    Bl2                Cache Number
ASCGRsv1  DS    Bl2                Unused
ASCGCOD   DS    Bl8                Cast-out directory reads issued
ASCGCODA  DS    Bl8                Cast-out directory - async
ASCGCODS  DS    Bl8                Cast-out directory - sync
ASCGCOU   DS    Bl8                Unlock cast-out locks issued
ASCGCOUA  DS    Bl8                Unlock cast-out locks - async
ASCGCOUS  DS    Bl8                Unlock cast-out locks - sync
ASCGDR    DS    Bl8                Directory reads issued
ASCGDRA   DS    Bl8                Directory reads issued - sync
ASCGDRS   DS    Bl8                Directory reads issued - async
ASCGPub   DS    (0*9)Bl8           Publishing requests
ASCGSync  DS     Bl8                 Update sync
ASCGXEnd  DS     Bl8                 BT/CL/ET transaction end
ASCGRedo  DS     Bl8                 Redo threshold
ASCGFull  DS     Bl8                 Full buffer pool
```

```
ASCGAll   DS     Bl8                 All blocks
ASCGRABN  DS     Bl8                 Specific RABN
ASCGDS    DS     Bl8                 File DS blocks
ASCGDSST  DS     Bl8                 DSST blocks
ASCGNI    DS     Bl8                 File NI blocks
*
ASChGL    Equ  *-ASChG
```

Adabas Global Lock Activity Section: LOCK

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Global lock statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DXLOCK operator command. There is one detail section instance for each lock type. The lock type is implied by the sequence number of the instance, starting with one.

```
ASLok     DSect ,                  Global Lock Section
*                                  Lock Types
ASLokGC   Equ     1                  GCB
ASLokSE   Equ     2                  Security
ASLokFS   Equ     3                  FST
ASLokUF   Equ     4                  UFT
ASLokSO   Equ     5                  Save Online
ASLokFL   Equ     6                  Flush
ASLokES   Equ     7                  Global ET Synchronization
ASLokRC   Equ     8                  Recovery
ASLokUT   Equ     9                  UFT-File
ASLokIU   Equ    10                  Index Update
ASLokHI   Equ    11                  Hold ISN
ASLokUD   Equ    12                  Unique DE
ASLokET   Equ    13                  ETID
ASLokLT   Equ    14                  LOB Tracker
ASLokCM   Equ    15                  Command Manager User
ASLokDI   Equ    16                  Data Increment
ASLokCP   Equ    17                  Checkpoint
ASLokDT   Equ    18                  Net-Work DBID Target Assignment
ASLokGU   Equ    19                  Global Update Commmand Sync
ASLokPM   Equ    20                  Parameter
ASLokDS   Equ    21                  DSF
ASLokRG   Equ    22                  RLOG
ASLokSP   Equ    23                  SPATS
ASLokCA   Equ    24                  Cancel
ASLokWR   Equ    25                  TBWK4A/E Table
ASLokWU   Equ    26                  PUTUA/E Table
ASLokXI   Equ    27                  XIDE
ASLokRH   Equ    28                  Replication Handshake
ASLokRI   Equ    29                  Read file/ISN
ASLokFA   Equ    30                  Format AC/AC1
ASLokOC   DS    Bl8                 Obtains  - Conditional
ASLokOG   DS    Bl8                 Obtains  - Granted
```

```
ASLokOR   DS    Bl8                Obtains  - Rejected
ASLokOU   DS    Bl8                Obtains  - Unconditional
ASLokOS   DS    Bl8                Obtains  - Synchronous
ASLokOA   DS    Bl8                Obtains  - Asynchronous
ASLokAC   DS    Bl8                Alters   - Conditional
ASLokAG   DS    Bl8                Alters   - Granted
ASLokAR   DS    Bl8                Alters   - Rejected
ASLokAU   DS    Bl8                Alters   - Unconditional
ASLokAD   DS    Bl8                Alters   - Deadlock/Rejected
ASLokAS   DS    Bl8                Alters   - Synchronous
ASLokAA   DS    Bl8                Alters   - Asynchronous
ASLokRL   DS    Bl8                Releases
ASLokRS   DS    Bl8                Releases - Synchronous
ASLokRA   DS    Bl8                Releases - Asynchronous
*
ASLokL    Equ   *-ASLok                                              ↩
```

Adabas Internucleus Messaging Control Block Activity Section: MSGB

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Internucleus messaging statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DXMSG operator command. This detail section appears with one instance. The number of blocks allocated (`ASMsgBBA`) and the high water mark (`ASMsgBBH`) reflect the entire nucleus session in interval records.

```
ASMsgB    DSect ,                  Inter-Nucleus Messaging Counts
ASMsgBBA  DS    Bl8                Message control blocks allocated
ASMsgBBH  DS    Bl8                Message control blocks used         +
                                    (high water mark)
ASMsgBBR  DS    Bl8                Message control block requests
*
ASMsgBL   Equ  *-ASMsgB
```

Adabas Internucleus Messaging Counts Section: MSGC

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Internucleus messaging statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DXMSG operator command. This detail section appears with one instance. Adabas Parallel Services nuclei report only the count of messages sent.

```
ASMsgC    DSect ,                  Inter-Nucleus Messaging Counts
ASMsgCMT  DS    Cl4                Message type
ASMsgCMS  DS    Bl8                Messages sent
ASMsgCMI  DS    Bl8                Messages incoming (arrived)
ASMsgCMA  DS    Bl8                Messages accepted
ASMsgCRS  DS    Bl8                Replies sent
*
ASMsgCL   Equ  *-ASMsgC
```

Adabas Internucleus Messaging Service Time Histogram Section: MSGH

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Internucleus messaging statistics are available only for Adabas Cluster Services and Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DXMSG operator command. This detail section appears with two instances:

1. The first represents messages subject to the MXMSG timeout parameter.
2. The second represents certain control messages not subject to MXMSG.

The two instances may be summed for a single representation of all messages. All message times are in microseconds. The minimum and maximum durations (`ASMsgHMn` and `ASMsgHMx`) reflect the entire nucleus session in interval records. Field `ASMsgMD2` is an extended (16-byte) floating point sum of the squares of all message durations. It may be used to compute a standard deviation.

```
ASMsgH    DSect ,                  Inter-Nucleus Messaging Histogram
ASMsgHXP  DS    Cl4                Transport service
ASMsgHMM  DS    Bl4                MXMSG or zero for messages not      +
                                    subject to MXMSG
ASMsgHMC  DS    Bl8                Message count
ASMsgHMD  DS    Bl8                Sum of all message durations
ASMsgHMS  DS    Bl16               Sum of squares, all msg durations   +
                                     (extended hex floating point)
ASMsgHMn  DS    Bl4                Minimum duration (us)
ASMsgHMx  DS    Bl4                Maximum duration (us)
ASMsgHCt  Equ   9                  Number of histogram buckets
ASMsgHG   DS    (0*ASMsgHCt)Bl8    Histogram buckets
ASMsgH10  DS     Bl8                 > 1000  s
ASMsgH09  DS     Bl8                 >  100  s, <= 1000  s
ASMsgH08  DS     Bl8                 >   10  s, <=  100  s
ASMsgH07  DS     Bl8                 >    1  s, <=   10  s
ASMsgH06  DS     Bl8                 >  100 ms, <=    1  s
ASMsgH05  DS     Bl8                 >   10 ms, <=  100 ms
ASMsgH04  DS     Bl8                 >    1 ms, <=   10 ms
ASMsgH03  DS     Bl8                 >  100 us, <=    1 ms
ASMsgH02  DS     Bl8                 <= 100 us
*
ASMsgHL   Equ  *-ASMsgH
```

Adabas Parallel Services Cache Activity Section: CSHP

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Parallel services cache statistics are available only for Adabas Parallel Services nuclei. They are derived from the ones presented at nucleus shutdown or in response to a DXCACHE operator command. This detail section appears with one instance. The directory high water mark `ASCPDHiN` and in-use count `ASCPDirI` reflect the entire nucleus session in interval records.

```
ASChP     DSect ,                  Parallel Services Cache Activity
ASCPCN    DS    Bl2                Cache Number
ASCPRsv1  DS    Bl2                Unused
ASCPNDir  DS    Bl8                Number of directory elements
ASCPNDiI  DS    Bl8                Number of directory index elements
*                                  Directory Statistics
*                                    General
ASCPDHiN  DS    Bl8                    High-water mark, this nucleus
ASCPDirI  DS    Bl8                    In-use, this nucleus
*                                    Read
ASCPDRA   DS    Bl8                    Located active
ASCPDRF   DS    Bl8                    Obtained from free pool
ASCPDRC   DS    (0*4)Bl8               Reclaim criteria categories
ASCPDNN   DS     Bl8                     First choice criteria
ASCPDND   DS     Bl8                     Second choice criteria
ASCPDIN   DS     Bl8                     Third choice criteria
ASCPDID   DS     Bl8                     Fourth choice criteria
ASCPDCF   DS    Bl8                    Unable to obtain (cache full)
ASCPDRT   DS    Bl8                    Tested for reclaim
*                                    Write
ASCPDWF   DS    Bl8                    Obtained from free pool
*                                  Space Management Statistics
*                                     Request Statistics
ASCPSRP   DS    Bl8                      Sufficient preallocated space
ASCPSRF   DS    Bl8                      Free space allocated
ASCPSRN   DS    Bl8                      Reclaim space, first choice
ASCPSRI   DS    Bl8                      Reclaim space, second choice
ASCPSRU   DS    Bl8                      Space unavailable (cache full)
ASCPSSP   DS    Bl8                      Searched part of space chain
ASCPSSF   DS    Bl8                      Searched entire  space chain
ASCPSST   DS    Bl8                      Number of space seqs tested
*                                     Element Reclaim Statistics
ASCPSEN   DS    Bl8                      First choice criteria
ASCPSEI   DS    Bl8                      Second choice criteria
*                                     Latch management statistics
*                                       Cache Space Chain
ASCPSPGE  DS    Bl8                        Get     Exclusive
ASCPSPWF  DS    Bl8                        WaitFor Exclusive
ASCPSPRE  DS    Bl8                        Release Exclusive
*                                       Cache Directory Index
ASCPDIGE  DS    Bl8                        Get     Exclusive
ASCPDIGS  DS    Bl8                        Get     Shared
ASCPDIUE  DS    Bl8                        Upgrade Exclusive
```

```
ASCPDIWE  DS    Bl8                       WaitFor Exclusive
ASCPDIWS  DS    Bl8                       WaitFor Shared
ASCPDIWU  DS    Bl8                       WaitFor Upgrade
ASCPDIRE  DS    Bl8                       Release Exclusive
ASCPDIRS  DS    Bl8                       Release Shared
*                                       Cache Directory
ASCPDRGE  DS    Bl8                       Get     Exclusive
ASCPDRGS  DS    Bl8                       Get     Shared
ASCPDRUE  DS    Bl8                       Upgrade Exclusive
ASCPDRWE  DS    Bl8                       WaitFor Exclusive
ASCPDRWS  DS    Bl8                       WaitFor Shared
ASCPDRRE  DS    Bl8                       Release Exclusive
ASCPDRRS  DS    Bl8                       Release Shared
*                                       Cast-Out Class
ASCPCOGE  DS    Bl8                       Get     Exclusive
ASCPCOGS  DS    Bl8                       Get     Shared
ASCPCOWE  DS    Bl8                       WaitFor Exclusive
ASCPCOWS  DS    Bl8                       WaitFor Shared
ASCPCORE  DS    Bl8                       Release Exclusive
ASCPCORS  DS    Bl8                       Release Shared
*
ASChPL    Equ  *-ASChP
```

ADARUN Parameter Value Section: PARM

This selectable detail section may appear in initialization records or whenever an ADARUN parameter is changed while the nucleus is running (subtypes 1 and 4 in the header section). It will not be generated for interval or termination records (subtypes 2 and 3). This section has a fixed-length portion containing most parameters, followed by variable-length areas for parameters capable of multiple values or lists of values.

Where possible, the individual field names are formed by prefixing the shortest allowable form of the parameter with ASP. In general, the SMF record will report character parameters in EBCDIC and numeric parameters in binary.

Parameters with limited enumerated values (YES or NO, for example) are reported in 1-byte fields if the possible values are unambiguous. Otherwise, the field length is that used by the nucleus, usually 4 bytes.

Here are some sample entries:

```
ASParm   DSect ,
ASPAO    DS    Cl1      AOslog
ASPARE   DS    Bl4      ARExclude
                           Offset to file table
ASPARMN  DS    Cl16     ARMname
ASPASS   DS    Cl1      ASSocache
ASPASY   DS    Cl1      ASYtvs
 . . .
```

```
ASPVI    DS    Cl1      Vista
ASPV64B  DS    Cl1      V64Bit
ASPWO    DS    Cl1      Workcache
ASParmV  DS    0B       Begin variable part
ASParmL  Equ   *-ASParm Length of ID section ↩
```

```
ASParm   DSect ,                   ADARUN Parameters
*
*         Adabas Nucleus Parameters
*
ASPAO    DS    Cl1                 AOslog
ASPARE   DS    Bl4                 ARExclude       Offset to file table
ASPARMN  DS    Cl16                ARMname
ASPASS   DS    Cl1                 ASSocache
ASPASY   DS    Cl1                 ASYtvs
 . . .

ASPVI    DS    Cl1                 VIsta
```

The ADARUN AREXCLUDE parameter is a variable length list of values. The base parameter entry will be an offset from the beginning of the detail section to the table of values. The table is a 4-byte inclusive field followed by 4-byte file numbers. A separate DSECT maps the AREXCLUDE file exclusion table:

```
ASPAFE   DSect ,                   ARM File Exclusion Table
ASPAFEN  DS    Bl4                 Inclusive length of table
ASPAFEF  DS    0Bl4                First file number entry
```

I/O by DD Name Section: IODD

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). I/O by DD data is derived from data presented at nucleus shutdown. There is one instance for each DD statement administered by the nucleus. You should examine the DD name and not rely on any observed order of the instances. An Adabas nucleus may open and close the same DD name multiple times. Multiple uses of a DD name are summed.

You might see this many DD statement entries in a single nucleus:

Statement Type	Number of DD Statements
ADC	248
ADP	248
ADW	62
ASSO	99
CLOG	8
DATA	99
ECS	1

Statement Type	Number of DD Statements
PLOG	8
RLOG	1
WORK	2

DD names beginning with "AD" represent files that were dynamically allocated, such as during Cluster or Parallel Services recovery, after another nucleus failed.

ADC and ADP represent CLOGs and PLOGs from other nuclei. There may be one set of 8 for each of the 31 possible other nuclei up to a maximum of 248.

Similarly, ADW represents a WORK data set from another nucleus. There may be two for each of the 31 possible other nuclei up to a maximum of 62.

```
ASIODD   DSect ,                    I/O Activity by DD
ASIODDNm DS    Cl8                  DD Name
ASIODDRd DS    Bl8                  Reads
ASIODDWt DS    Bl8                  Writes
*
ASIODDL  Equ   *-ASIODD
```

Session Statistics Section: SESS

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Session statistics data is similar to the data presented at nucleus shutdown or at the DSTAT operator command.

There is one single instance per record. The section has a fixed-length.

Here is a shortened sample of the SESS section:

```
ASSess   DSect ,                   Nucleus session statistics
*
ASSDURA  DS    BL8                 Nucleus Session Duration
ASSWAIT  DS    BL8                 Nucleus Wait Time
ASSCPU   DS    BL8                 Nucleus CPU Time

ASSASR   DS    BL8                 ASSO Reads
ASSASW   DS    BL8                 ASSO Writes

...

ASSBFFE  DS    BL4                 Buffer efficiency .x precision
ASSessL  Equ  *-ASSess            Length of the session section
```

Storage Pool Section: STG

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Storage pool statistics are derived from statistics presented at nucleus shutdown or in response to a DRES operator command. There is one instance for each storage pool with a non-zero size. Be sure to examine the pool name and not rely on any observed order of the instances.

Storage pool statistics are reported two ways: in bytes and also in units such as a user might specify as an ADARUN parameter, for example, NC. When the units are bytes, the two sets of statistics are the same.

Normally an interval record would show the change from the previous interval, but that isn't meaningful for storage pools. Thus the interval and termination record subtypes all reflect total usage for the nucleus session.

```
ASStg    DSect ,                    Storage Pool Usage
ASStgNm  DS    Cl4                  Storage pool name
ASStgBSz DS    Bl8                  Size in bytes
ASStgBHW DS    Bl8                  High water mark in bytes
ASStgUSz DS    Bl8                  Size in units from ADARUN parameter
ASStgUHW DS    Bl8                  High water mark in ADARUN units
ASStgHWT DS    Bl4                  High water mark timestamp
*
ASStgL   Equ   *-ASStg              Length of Storage Pool section
```

These are the possible storage pools:

Storage Pools		
Pool Name	**ADARUN Parameter**	**Description**
AB	NAB	Attached buffers
CQ	NC	Command queue
DUQ	LDEUQP	Unique descriptor
FI	LFP	Internal format buffers
HQ	NH	Hold queue
PLIO	NPLOGBUFFERS	PLOG I/O buffers
REDO	LRDP	Deferred publishing
RPL	LRPL	Replication pool
SC	LCP	Security information
TBI	LI	ISN table
TBS	NQ	Sequential command table
UQ	NU	User queue element
UQF	NU	User queue file elements
WKIO	NWORK1BUFFERS	Work I/O buffers

Storage Pools		
Pool Name	**ADARUN Parameter**	**Description**
WORK	LWP	Work
WKP1	LP	Work part 1
WK1B	CLUWORK1CACHE	Work part 1b
WKP2	LWKP2	Work part 2
WKP3	--	Work part 3
XID	NU	Transaction ID

Thread Activity Section: THRD

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). Thread activity data is derived from data presented at nucleus shutdown or in response to a DTH operator command. The ADARUN parameter NTHREAD defines the number of user threads for the nucleus session. There is one instance for each defined user thread. The thread number is implied by the sequence number of the instance.

```
ASThrd   DSect ,                    Thread Activity
ASThrdCt DS     Bl8                 Number of commands executed in this +
                                      thread
*
ASThrdL  Equ   *-ASThrd
```

zIIP Statistics Section: ZIIP

This selectable detail section may appear in interval or termination records (subtypes 2 and 3). zIIP activity data is derived from data presented at nucleus shutdown or in response to a DZSTAT operator command. There is one instance for each zIIP-enabled nucleus product.

```
ASzIIP    DSect ,                   Nucleus session statistics
*
ASzProd   DS    Cl4                 Sub-product name
ASzPNuc    Equ   c'ANUC'             Adabas nucleus main task
ASzPRevL   Equ   c'REVL'             Review local
ASzRsv1   DS    Bl4                 Unused
*
ASz2SRB   DS    Bl8                 Count of switches to SRB mode
ASz2TCB   DS    Bl8                 Count of switches to TCB mode
ASzPAUT   DS    Bl8                 Count of pause TCB requests
ASzRLST   DS    Bl8                 Count of release TCB requests
ASzPAUS   DS    Bl8                 Count of pause SRB requests
ASzRLSS   DS    Bl8                 Count of release SRB requests
ASzQPRQ   DS    Bl8                 Count of queue parallel requests
ASzXRQE   DS    Bl8                 Count of no free RQE for requests
*
ASzNCtr   Equ   64                  Number of event counters
```

```
ASzCtr    DS    (ASzNCtr)Bl8       Event counters
*
ASzCPUT   DS    Bl8                CPU time             (STCK format)
ASzZTim   DS    Bl8                zIIP time            (STCK format)
ASzSCPU   DS    Bl8                SRB-on-CPU time      (STCK format)
ASzZQCP   DS    Bl8                zIIP-qualified time  (STCK format)
*
ASzIWait  DS    Bl8                ADAIOR general WAITs
ASzIRlse  DS    Bl8                ADAIOR releases
*
ASzIIPL   Equ  *-ASzIIP            Length of the zIIP section
```

ASMFREC Mapping Macro

Use the ASMFREC macro to generate the latest SMF record DSECTs. The ASMFREC macro will always generate the header and self-defining section DSECT. Detail section DSECTs will be generated as specified. The header and self-defining sections are mapped by a single DSECT. Each detail section is mapped by its own DSECT. The syntax of the ASMFREC macro is:

```
label    ASMFREC   Prefix={AS | prefix},
                                 Detail={All | (type [,type]...)},
                                 Title = {'Adabas SMF Records' | 'string'}
```

Prefix

Specify a character string to be used as the initial characters for all DSECT and field names. The default is Prefix=AS.

Detail

Identify which detail section DSECTs are to be included in the expansion. "All" is the default and will include all detail sections. Alternatively, a comma-delimited list of types (enclosed in parentheses) can be specified; only the types specified will be included. The valid types are shown in the following table. A null value (Detail=) will inhibit all detail section DSECTs.

ASMFREC Macro Specification	Detail Section Description	ASMFREC DSECT Name Produced by the ASMFREC Macro[1]
CMD	Adabas command activity	*xx*Cmd
CSHB	Adabas global cache activity by block type	*xx*ChB
CSHF	Adabas global cache activity by Adabas file number	*xx*ChF
CSHG	Adabas global cache activity	*xx*ChG
CSHP	Adabas Parallel Services cache activity	*xx*ChP
FILE	Adabas file activity	*xx*File

ASMFREC Macro Specification	Detail Section Description	ASMFREC DSECT Name Produced by the ASMFREC Macro[1]
LOCK	Adabas global lock activity	*xx*Lok
MSGB	Adabas internucleus messaging control block activity	*xx*MsgB
MSGC	Adabas internucleus messaging counts	*xx*MsgC
MSGH	Adabas internucleus messaging service time histogram	*xx*MsgH
PARM	ADARUN parameter values	*xx*Parm
IODD	I/O by DD name	*xx*IODD
SESS	Nucleus session statistics	*xx*Sess
STG	Storage pool	*xx*Stg
THRD	Thread activity	*xx*Thrd
ZIIP	zIIP statistics	*xx*zIIP
USER	User-defined	user-defined

Where *xx* is the prefix specified in the ASMFREC macro.

Title

If the Title default "Adabas SMF Record " or another quoted string is specified, an assembler Title statement is generated before the header section DSECT. A null value (Title=) for this operand will inhibit a title in the DSECT.

SMF User Exit

You can provide a user exit if you want to add a detail section to the Adabas SMF record. The user exit is a separate load module whose name must be provided in the ADARUN UEXSMF parameter. For complete information about the user exit, read *SMF User Exit*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*.

IBM Type 89 SMF Records

An Adabas nucleus can register with z/OS to have CPU usage statistics included in IBM type 89 SMF records. These records are described in *z/OS MVS System Management Facilities (SMF), IBM document SA22-7630*.

To activate type 89 recording for Adabas, specify ADARUN parameters SMF=YES and SMF89=YES. During initialization Adabas will register the nucleus address space with z/OS SMF and have its CPU statistics included in subtype 1 of the type 89 records. The address space is deregistered at nucleus termination. Each Adabas nucleus appears as a separate type 89 entry.

The type 89 entries include CPU usage and a number of descriptive registration parameters. Adabas nuclei use these descriptive fields in type 89 entries as follows:

SMF Type 89 Descriptive Fields				
Name	**Length**	**Format**	**Description**	**Value**
SMF89UPO	16	EBCDIC	Product owner or vendor name	SOFTWARE AG
SMF89UPN	16	EBCDIC	Product name	ADABAS
SMF89UPV	8	EBCDIC	Product version	The eight-byte product version has two-byte numeric values for the Adabas version, release, SM level, and cumulative level.
SMF89UPQ	8	Binary	Product qualifier	The product qualifier is a seven-byte string that may be used to distinguish among several nucleus instances. It contains a series of binary fields: **SVC** (1 byte) **DBID** (4 bytes) **NucID** (2 bytes) Use both the SVC and DBID to identify instances of Adabas Cluster or Parallel Service nuclei for the same database on any one system.
SMF89UPI	8	EBCDIC	Product ID	The product ID is a string of up to eight single characters to show what add-on products are being used. The characters may appear in any order: **C** (Adabas Cluster Services) **D** (Adabas Delta Save) **F** (Adabas Fastpath) **M** (Adabas Review) **P** (Adabas Parallel Services) **R** (Event Replicator for Adabas) **S** (Adabas Cache Facility) **T** (Adabas Transaction Manager)

SMF Type 89 Descriptive Fields				
Name	**Length**	**Format**	**Description**	**Value**
				U (Adabas Security) **V** (Adabas Vista) The following applies: C, D, P, and R are mutually exclusive. D, F, M, T and V are exclusive with A.

14 AFPLOOK

- Enabling AFPLOOK 222
- Operational Defaults 222
- Adjusting the Defaults 222
- AFPLOOK Parameters 223
- AFPLOOK Report 225

The command sampler (AFPLOOK) can be used to determine where the best results may be expected from Fastpath by reporting on the command constructs that qualify for Fastpath.

The sampler can be controlled and viewed online using SYSAFP but may also be controlled without SYSAFP using the defaults module AFPLUKD. In either case, the sampler will print a report in DDPRINT at ADAEND time if a sample is active at that time.

Enabling AFPLOOK

AFPLOOK is enabled using the following ADARUN command:

```
ADARUN FASTPATH=YES
```

Use the SYSAFP administration center to activate the sample.

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 ***AFPLOOK Parameters***.

Adjusting the Defaults

AFPLOOK is designed for dynamic use. It is inactive by default. You use the SYSAFP administration center to activate and target it. However, you can control it independently of the administration center.

It is possible to configure AFPLOOK independently of SYSAFP by creating a defaults module AFPLUKD using the macro AFPMLUKD supplied on the ADA*vrs*.SRCE library. Please refer to the sample job SAGLUKD supplied on the ADA*vrs*.JOBS library.

Each of the parameters described in the section **AFPLOOK Parameters.** can be pre-configured using AFPLUKD.

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
- Demo Sampling (AFPLUKD Only)
- Real Sampling (AFPLUKD Only)
- Report Title (AFPLUKD only)

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

Demo Sampling (AFPLUKD Only)

For use when pre-configuring AFPLOOK with the defaults module AFPLUKD and licensed Fastpath is not present.

`DEMO=` sets whether demo AFPLOOK samples by default or not.

Settings are: ON / OFF

Default: ON

Real Sampling (AFPLUKD Only)

For use when pre-configuring AFPLOOK with the defaults module AFPLUKD and licensed Fastpath is present.

`REAL=` sets whether real AFPLOOK samples by default or not.

Settings are: ON / OFF

Default: OFF

Report Title (AFPLUKD only)

For use when pre-configuring AFPLOOK with the defaults module AFPLUKD.

This parameter sets the title that will be seen on the output report. This can be a maximum of 30 characters.

Default: AFPLOOK report summary

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 Fastpath. The actual optimization will depend on various factors unique to each user site. Contact Software GmbH 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
-----------------------------------------------------------------
```

15 AVILOOK

- Enabling AVILOOK 230
- SYSAVI – Selecting AVILOOK 230
- SYSAVI – Using AVILOOK 231

The 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 media.

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..: _____ Nuc 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  2006-04-20 |
|                                        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 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   2006-04-20 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: 2006-04-20 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 Add Files

To add a new AVILOOK file

1 Press PF10 from the AVILOOK File Maintenance screen.

```
+-------------------------------------------------------------------------+
|  08:11:07              AVILOOK Add File                       2006-04-20 |
|                                                                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.

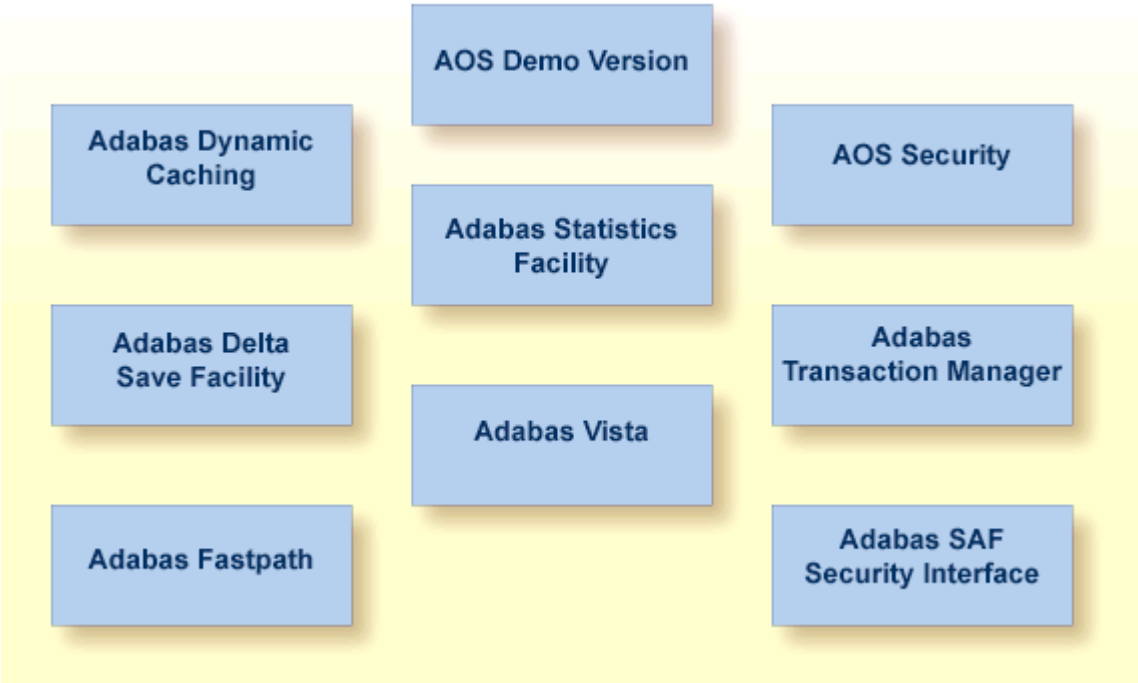
5 Press PF5 to add the file.

Note: Files defined (and activated) are only sampled for the duration of the current database session.

16 Adabas Online System Demo Version

- Overview 236
- Main Menu Functions 238
- Session Monitoring 240
- List Checkpoints 252
- File Maintenance 255
- Database Maintenance 257
- System Operator Command Functions 258
- Database Report 265

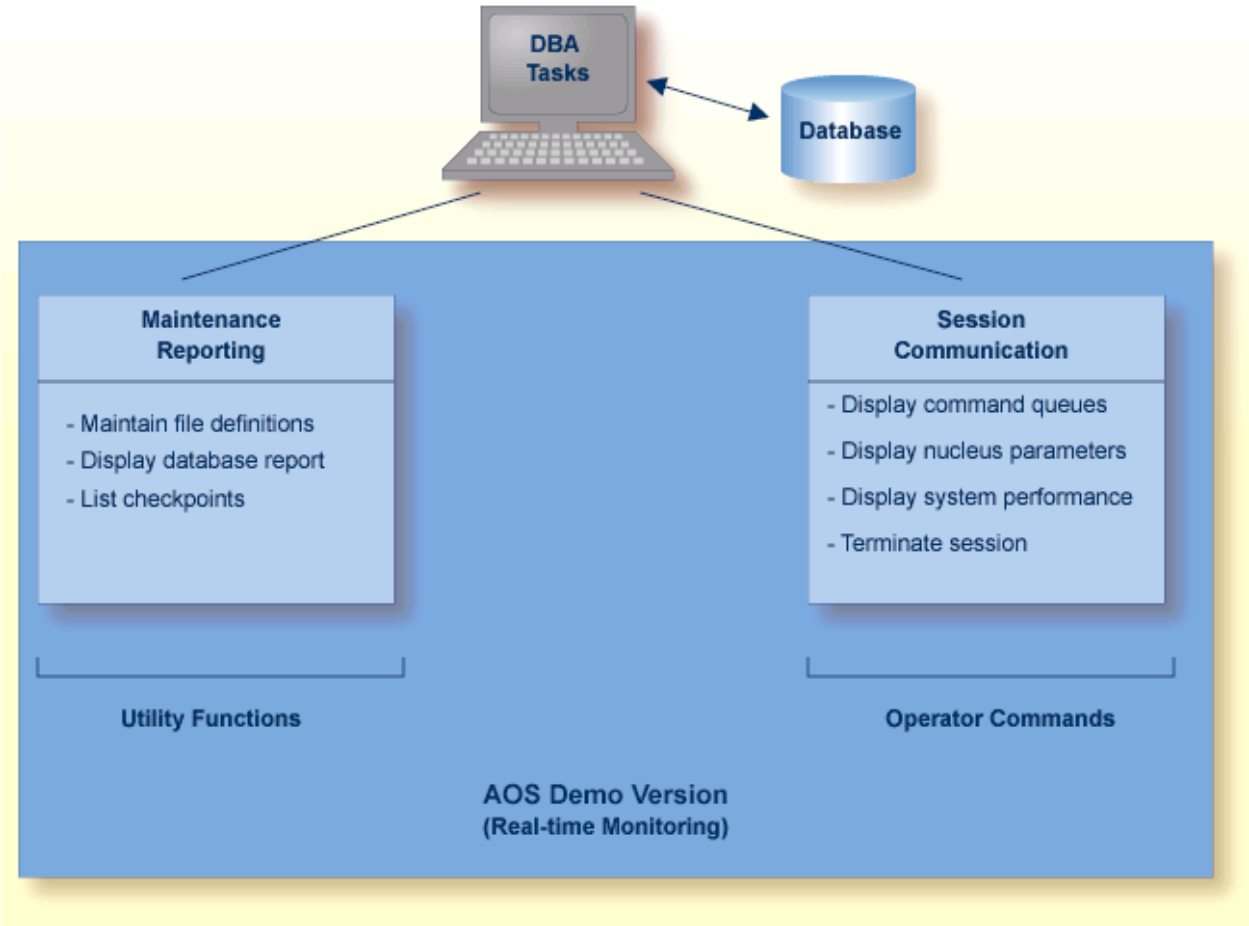
A demo version of Adabas Online System (AOS) and access to the online services for selected other Adabas products and facilities is included with Adabas as shown in the following diagram:



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.

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:



See the Adabas Installation documentation for information about installing the demo version of AOS.

The AOS add-on product includes services that correspond to additional utility functions and operator commands. Read the Adabas Online System documentation for information.

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).

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.

```
12:59:18          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
Cluster                        -  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:

Option	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)
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.

- Specifying the AOS Demo Version Database
- Using Program Function (PF) Keys
- Selecting a Menu Option
- Getting Help
- AOS Demo Version Messages

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 (ADARSP022) 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.

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.

```
13:10:35          ***** A D A B A S  BASIC  SERVICES *****          2014-04-29
                            -  Session Monitoring  -                PAC0012


 Code   Service                          Code   Service
 ----   ---------------------------      ----   ----------------------------
  *     Display cluster members           *     Current snapshot statistics
  *     Maintain user profiles            *     Maintain TCP/IP URL
  D     Display parameters                U     Display interval utilization
  I     Display installed products        *     Replicator Management
  *     Display event log buffer          *     Cluster usage
  *     Modify parameters                 Z     Display maintenance levels       ↩

  Q     Display queues
  *     Refresh interval statistics
  ?     Help                              .     Exit
 ----   ---------------------------      ----   ----------------------------
        Code ......... _
        Database ID .. 1955    (WIS1955)               NucID .. 1021


Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   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:

Option	Function
D	Display Adabas nucleus (ADARUN) parameters
I	Display a list of installed products
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

- Display ADARUN Parameters
- Display Installed Products
- Display Hold Queue
- Display System Status and Thread Usage

- Display Maintenance Levels

Display ADARUN Parameters

You can view Adabas nucleus (ADARUN) parameters.

To view the parameters, select option D from the Session Monitoring menu.

Multiple screens are used for displaying parameters:

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                  -  Display Parameters  -                 PACPD22
NucID 1021

 -------------- Pools --------------  ------------- Queues -----------------
 Sort Area         (LS).. 19968       Command Queue          (NC) .. 20
 Int. User Buffer  (LU).. 400000      Hold Queue             (NH) .. 800
 Buffer Pool      (LBP).. 80896       User Queue             (NU) .. 200
 Format Pool      (LFP).. 150000      ------------ Time Windows -------------
 ISN List Table    (LI).. 360000      Transaction Time       (TT) .. 4858
 Seq. Cmd. Table   (LQ).. 20000       Max Transaction Time (MXTT) .. 3600
 Work Pool        (LWP).. 1500000     Nonactivity ACC-User (TNAA) .. 4858
 Attached Buffer  (NAB).. 100         Nonactivity ET-User  (TNAE) .. 4858
 Security Pool    (LCP).. 10000       Nonactivity EXU-User (TNAX) .. 4858
 UQ-DE Pool    (LDEUQP).. 50000       Max Nonactivity Time(MXTNA) .. 3600
 Err. Recovery (MSGBUF).. 36          Time Limit Sx-Cmds (TLSCMD) .. 300
                                      Max Time for Sx-Cmds(MXTSX) .. 3600
                                      Command Time           (CT) .. 3858
                                      SYNS60 Interval    (INTNAS) .. 3600

                                                                  Page 1 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify            +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

--------- Miscellaneous -----------     --------- Miscellaneous --------------
Read only     (READONLY).. NO           LARGEPAGE ................. NO
UTI only       (UTIONLY).. NO           V64BIT .................... NO
OPEN required   (OPENRQ).. NO           SRLOG ..................... Upd
Ignore DIB Entry(IGNDIB).. NO           LOGWARN ................... 0
Local nucleus   (LOCAL) .. NO           SECUID .................... ALLOW
Number of Threads  (NT) .. 25           INDEXCROSSCHECK ........... YES
Non DE Search  (NONDES) .. YES          Print on refresh(REFSTPRT). YES
Log AOS/DBS Upd.(AOSLOG).. NO           Number plog buffers ....... 1
Batch Support   (BATCH) .. NO           Number work1 buffers ...... 1
Data Protection Area(LP).. 1000         Event log buffer size ..... 1024
Ignore Wrk Prt4(IGNDTP) .. NO
WORK-Part-4 Area (LDTP) .. 0
WORK-Part-2 Area(LWKP2) .. 158
SVC                   (SVC) .. 249

                                                                  Page 2 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -        +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

--------- Space Warning ----------------
ASSOSPACEWARN ..... 0
DATASPACEWARN ..... 0

--------- Auto Increase ------------------------------------------------
AUTOINCASSOTHRESHOLD ..... 0          AUTOINCDATATHRESHOLD ..... 0
AUTOINCASSOSIZE ... 0          CYL    AUTOINCDATASIZE ... 0          CYL
AUTOINCASSOTOTAL .. 0          CYL    AUTOINCDATATOTAL .. 0          CYL

--------- SMF --------------------------
SMF ............... NO





                                                                  Page 3 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -        +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

 ---- Command Logging ----      ------------ Command Logging --------------
 Command Logging .. YES         Log VOLSER info    (LOGVOLIO) .. NO
 LOGCB ............ NO          Max buffer size/cmd (CLOGMAX) .. 16384
 LOGFB ............ YES         Max buffer size/buf(CLOGBMAX) .. 4096
 LOGRB ............ YES         Log ABDX            (LOGABDX) .. NO
 LOGSB ............ NO          Log multifetch buffer (LOGMB) .. NO
 LOGVB ............ NO          Command log layout(CLOGLAYOUT).. 5
 LOGIB ............ NO
 LOGIO ............ NO
 LOGUX ............ NO
 LOGSIZE .......... 5064        ---------- Protection Logging -----------
 DUAL CLOG Size ... 675         PLOG required        (PLOGRQ) .. YES
 DUAL CLOG Dev. ... 3390        DUAL PLOG Size      (DUALPLS) .. 240
 NCLOG ............ 0           DUAL PLOG Device    (DUALPLD) .. 3390
                                NPLOG ......................... 0

                                                                  Page 4 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -        +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

------------ Large Pools -------------      --------- Other Services ---------
Flush I/O Pool (LFIOP)  .. 80000            Triggers / Procedures (SPT) .. NO
                                            Delta Save Facility   (DSF) .. YES
                                            Cache Facility      (CACHE) .. NO
                                            Transaction Manager   (ATM) .. NO
-------- User Specific Limits --------      TCP/IP Support      (TCPIP) .. NO
Hold Queue Limit  (NISNHQ) .. 200           Ext. Error Recovery  (SMGT) .. YES
CIDs per User      (NQCID) .. 40            2 Phase Commit Support(DTP) .. NO
ISN per TBI Element(NSISN) .. 100
                                            Review:
------------ Buffer Pool -------------       Support  (REVIEW) .. LOCAL
Bufferflush Dur.  (TFLUSH) .. 1
Parallel LFIOP I/O (FMXIO) .. 1              Filter ............. YES
Async. by Vol-Ser (ASYTVS) .. YES            Max bufsize cmd .... 16384
                                             Max bufsize buf .... 5120

                                                                  Page 5 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -        +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

----- Cluster/Parallel Services -------    ---- Cluster/Parallel Services ----
Environment .......... Sysplex             MXMSG .................. 300
ARM element name .....                     MXMSGWarn .............. 0
Cache structure name . ADA_CACHE4          MXCANCEL ............... 300
Lock structure name .. ADA_LOCK4           MXCANCELWarn ........... 75
Sysplex group name ... WISPLEX             MXWtor ................. 0
Cache type ........... DSP                 MXStatus ............... 15
DIrratio ............. 4
ELemratio ............ 1
Redo Pool (LRDP) ..... 80000
CLOGMRg .............. NO
CLUCACHEUnchanged .... No
CLULOCKSize .......... 0
CLUCACHESize ......... 0


                                                                  Page 6 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -        +        Menu        ↩
```

```
14:03:49          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                   -  Display Parameters  -                 PACPD22
NucID 1021

---- Replication Parameters -------
Replication .......... YES
RPWARNPercent ........ 0
RPWARNINCrement ...... 10
RPWARNINTerval ....... 60
RPWARNMessagelimit ... 5
RPCONNECTCount ....... 0
RPCONNECTInterval .... 0
RPLSORT .............. YES
LRPL ................. 100000









                                                                  Page 7 of 7
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   Exit      ABC List  Modify   -                 Menu        ↩
```

Display Installed Products

Choose option **I** on the Session Monitoring menu and press ENTER to display a list of installed products.

```
14:06:21          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBid  1955             - Display Installed Products -               PACII02
NucID 1021

   -------------------------- -----    ------------------------------- ----
   Cache Facility ........... NO       Extended Error Recovery ....... YES
   Delta Save Facility ...... YES      Recovery Aid .................. YES
   Cluster Services ......... YES      Stored Procedures & Triggers .. NO
   Parallel Services ........ NO       Two Phase Commit .............. NO
   Fastpath ................. NO       TCPIP support ................. NO
   Vista .................... NO       Event Replicator .............. YES
   Transaction Manager ...... NO
   SAF Security Interface ... NO
   Review ................... LOCAL
   Adabas Online System ..... YES





Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu        ↩
```

This screen displays what is installed on the current selected Adabas.

Display Hold Queue

Selecting *Queue displays* (option Q) from the Session Monitoring menu displays the following menu:

```
14:07:11          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                            -  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 ........ 1955      (WIS1955)             NucID .. 1021

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   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 Interval Utilization menu:

```
14:08:04          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                          -  Interval Utilization  -                PACU002

Code  Service                            Code  Service
----  -------------------------------    ----  -------------------------------
 *    Command usage                       S    System status
 *    File usage                          T    Thread usage
 *    High watermarks (pools/queues)      *    I/Os by Volser
 *    Workpool (LWP) usage                *    WORK status
 *    Nucleus File Status                 *    Display PPT table
 *    PLOG status
 ?    Help
 .    Exit
----  -------------------------------    ----  -------------------------------


        Code ......... _
        File Number .. 0
        Database ID .. 1955   (WIS1955)            NucID .. 1021

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help     NextNuc   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 data sets; remote and local call distribution; and other current session status information.

```
14:10:32          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                      -  System Status  -                  PACUS12
NucID: 1021

 Physical    Reads          Writes                Call Distribution
        ----------------------------  ---------------------------------------
 ASSO         1,106             124    No. of HQEs active .........          0
 DATA         1,416              39    No. of UQEs in User Queue ..          1
 WORK             2             161    No. of CQEs waiting in CQ ..          0
 PLOG            15             120    No. of PLOG switches .......          0
 CLOG             9               6    No. of CLOGs ...............          2
                                       No. of PLOGs ...............          2
 PLOG protection blks .         119
 PLOG protection I/Os .         119    Format translations .....          45
 WORK1 protection blks.         161    Format overwrites .......           0
 WORK1 protection I/Os.         161    Auto-restarts ...........           2
                                       Throw-backs for ISN .....           0
 Logical reads ........      12,140    Throw-backs for Space....           0
 Buffer efficiency ....         4.8
                                                                  page 1 of 3

PF1---- PF2---- PF3----- PF4----- PF6----- PF7----- PF8----- PF9----- PF12----
Help            Exit     Refresh                     +        FullView Menu    ↩
```

```
14:10:32          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                      -  System Status  -                  PACUS12
NucID: 1021

         Call Distribution                          Commands
-------------------------------------      ------------------------------
User sessions ........          22        Remote ........           0
Buffer flushes .......          39        Local .........       2,861
Flush phases .........          39        Internal ......       2,824
Blocks flushed .......         156        Operator ......           0
Flush I/Os ...........         156        Total .........       5,685


       Flush Requests Returned                     Time Elapsed
-------------------------------------      ----------------------------------
Immediately ..........           0        Duration .... 1 day, 17:36:50
After logical flush ..           0        Wait-time ... 1 day, 17:36:44
After entire flush ...          41        CPU time .... 0 day, 00:00:17

                                                                  page 2 of 3

PF1---- PF2---- PF3----- PF4----- PF6----- PF7----- PF8----- PF9----- PF12----
Help            Exit     Refresh            -        +        FullView Menu    ↩
```

```
14:10:32          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                     -  System Status  -                   PACUS12
NucID: 1021

                              Nucleus Status Flags
                      ------------------------------------
                      Adabas operation normal

                                                                  page 3 of 3

PF1---- PF2---- PF3----- PF4----- PF6----- PF7----- PF8----- PF9----- PF12----
Help            Exit     Refresh           -                 FullView 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.

```
14:12:25          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                      -  Thread Usage  -                   PACUT12
NucID 1021
Nr. I Thread Status     I Command Type    I Wait Event        I Nr. CMDs
-------------------------------------------------------------------------------
  1 I Active            I Simple Cmd.     I                   I       5,707
  2 I Not active        I                 I                   I          10
  3 I Not active        I                 I                   I           0
  4 I Not active        I                 I                   I           0
  5 I Not active        I                 I                   I           0
  6 I Not active        I                 I                   I           0
  7 I Not active        I                 I                   I           0
  8 I Not active        I                 I                   I           0
  9 I Not active        I                 I                   I           0
 10 I Not active        I                 I                   I           0
 11 I Not active        I                 I                   I           0
 12 I Not active        I                 I                   I           0
 13 I Not active        I                 I                   I           0
 14 I Not active        I                 I                   I           0
 15 I Not active        I                 I                   I           0

PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF9----- PF12-----
Help               Exit      Refresh                     Fullview Menu       ↩
```

Display Maintenance Levels

Selecting *Display maintenance levels* (option Z) from the Session Monitoring menu displays information about the Adabas nucleus modules:

```
14:13:25          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955             -  Display Maintenance Levels  -              PACZ002
NucID .. 1021

 Select Module Name:  ________
 ------------------------------------------------------------------------------
   ADARUN   RUNMVS   Date 2013-09-17, Version 8.3, SP 1
            RUNIND   Date 2014-03-21, Version 8.3, SP 1
   ADANCX            Date 2014-04-21, Version 8.3, SP 1
   ADAXCF            Date 2012-04-19, Version 8.3, SP 1
   ADAXEC            Date 2012-10-11, Version 8.3, SP 1
   ADAXEL            Date 2013-10-21, Version 8.3, SP 1
   ADACLU            Date 2014-04-15, Version 8.3, SP 1
   ADAMXI            Date 2013-12-10, Version 8.3, SP 1
   ADAMIM            Date 2014-03-21, Version 8.3, SP 1
   ADARVU            Date 2014-03-21, Version 8.3, SP 1
   ADACLX            Date 2013-12-30, Version 8.3, SP 1
   ADARMT            Date 2014-03-19, Version 8.3, SP 1


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.

List Checkpoints

Selecting *Checkpoint maintenance* (option C) from the Main Menu invokes the Checkpoint Maintenance menu:

```
14:14:45          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                         -  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:

Type	Description
SYNC	nucleus initialization
SYNF	user open EXF
SYNP	utility, without nucleus
SYNS	ADARES
SYNV	volume ID change
SYNX	utility
SYN1	ADASAV DB begin
SYN2	ADASAV DB begin
SYN4	ADASAV file begin

Type	Description
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:

```
14:15:13          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
DBID 1955                      - List Checkpoints -                  PCPC012

  CP   CP     Date      Time      PLOG     Block      Vol/Ser   User Job Name
 Name Type                        Number    Number      Number   Type
 ---- ---- ---------- --------   ------  ----------   --------  ---- --------
 SYNC  01  2010-12-14 11:29:34                                        USAWISCR
 SYNS  5B  2010-12-14 11:29:34                                   EXU  ADAEND
 SYNP  07  2010-12-14 11:30:19        2           1   DUAL           USAWISRE
 SYNC  01  2010-12-14 11:30:45        2           2   DUAL           USAWISRP
 SYNS  61  2010-12-14 11:32:40        2         417   DUAL           ADABAS
 SYNS  5B  2010-12-14 11:42:43        2       29341   DUAL      EXU  ADAEND
 SYNC  01  2010-12-14 11:43:26        3           1   DUAL           USAWISRP
 SYNS  60  2010-12-14 16:05:31        3        1524   DUAL           ADABAS
 SYNS  60  2010-12-15 10:08:02        3        1525   DUAL           ADABAS
 SYNS  60  2010-12-15 11:53:37        3        1526   DUAL           ADABAS
 SYNS  60  2010-12-15 12:56:37        3        1527   DUAL           ADABAS
 SYNS  60  2010-12-15 13:59:37        3        1528   DUAL           ADABAS
 SYNS  60  2010-12-15 15:02:37        3        1529   DUAL           ADABAS
 SYNS  60  2010-12-15 16:05:37        3        1530   DUAL           ADABAS
 SYNS  60  2010-12-15 17:08:42        3        1531   DUAL           ADABAS
PF1----- PF2------ PF3------ PF4------ PF6----- PF7 ----- PF8----- PF12-----
Help               Exit                Top      -        +        Menu       ↩
```

This screen illustrates an extended checkpoint list providing additional information about each checkpoint:

```
18:58:21          ***** A D A B A S  BASIC  SERVICES *****       2009-08-13
DBID 1955                      - List Checkpoints -                  PCPC012

  CP   CP     Date      Time      PLOG     Block      Vol/Ser   User Job Name
 Name Type                        Number    Number      Number   Type
 ---- ---- ---------- --------   ------  ----------   --------  ---- --------
 SYNP  30  2009-02-17 19:07:59                                        USAWISN0
      LOAD                FNR= 1
 SYNP  30  2009-02-17 19:07:59                                        USAWISN0
      LOAD                FNR= 2
 SYNP  30  2009-02-17 19:08:00                                        USAWISN0
      LOAD                FNR= 3
 SYNP  30  2009-02-17 19:08:01                                        USAWISN0
      LOAD                FNR= 6
 SYNP  30  2009-02-17 19:08:01                                        USAWISN0
      LOAD                FNR= 7
 SYNC  01  2009-02-17 19:08:02                                        USAWISN0
      SESSION OPEN        IGNDIB = N , FORCE = N
 SYNS  5B  2009-02-17 19:08:02                                   EXU  ADAEND
      REFRESH STATS
 SYNP  30  2009-02-17 19:17:04        2           1   DUAL           USAWISTA
PF1----- PF2------ PF3------ PF4------ PF6----- PF7 ----- PF8----- PF12-----
Help               Exit                Top      -        +        Menu        ↩
```

File Maintenance

Selecting option F from the Main Menu invokes the File Maintenance menu:

```
14:16:42          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                            -  File Maintenance  -                  PFL0004

Code  Service                           Code  Service
----  ------------------------------- ----  --------------------------------
  C   Define/modify FDT                   *   Modify file parameters
  *   Release descriptor                  *   Reorder file online
  *   Delete existing file                *   Refresh file to empty status
  *   Define new file                     *   Allocate/deallocate file space
  *   Logically delete/undel descriptr    *   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:

```
14:18:02          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                   -  FDT/SDT Definition / Modification  -          PFLC004

  Code  Service                           Code  Service
  ----  ----------------------------     ----  ----------------------------
    *   Add new field(s)                   *   Online invert
    *   Change field parameters            *   Define/add SDT
    D   Define new FDT                     *   Release UQ from descriptor
    *   Delete field from FDT              ?   Help
    *   Undelete field from FDT            .   Exit
  ----  ----------------------------     ----  ----------------------------
        Code ......... _
        File No. .....
        Field Name ... __
        File Password.
        Database ID .. 1955   (WIS1955)




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:

```
14:19:27          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                        -  Define FDT  -                   PFLCD12

File Number .... 55                            New FDT ... Y

Enter Field Description(s) ::

I Levl I Name I Length I Format I Options        I Date/time and System fields
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     Def SDT   Exit      Def File  Disp FDT                   Menu       ↩
```

FDTs for existing files cannot be redefined with this option.

This function corresponds to the Adabas utility function ADACMP COMPRESS.

Database Maintenance

Selecting option M from the Main Menu invokes the Database Maintenance menu:

```
14:20:01          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                          -  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. ...... 55
       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.

System Operator Command Functions

Selecting *Session opercoms* (option O) from the Main Menu displays the following menu:

```
19:36:22          ***** A D A B A S  BASIC  SERVICES *****          2009-08-18
                             -  Session Opercoms  -                 PACI002

 Code  Service                            Code  Service
 ----  ------------------------------     ----  ------------------------------
  *    Allocate/Deallocate CLOG/PLOG       S    Stop user(s)
  *    Issue reactivate CLOG command       T    Termination Commands
  E    Extended Error Recovery             *    Manage Online Utilities
  *    Force CLOG or PLOG switch           *    User Table Maintenance
  L    Lock or unlock files                *    Replicator Management
  *    Reset ONLINE-DUMP-Status
  .    Exit                                ?    Help
 ----  ------------------------------     ----  ------------------------------
       Code ........... _
       Userid(ETID) ... ________
       CLOG/PLOG Ind .. _
       Database ID .... 1955   (WIS1955)



Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu        ↩
```

```
14:44:58          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                             -  Session Opercoms  -                 PACI002

 Code  Service                            Code  Service
 ----  ------------------------------     ----  ------------------------------
  *    Allocate/Deallocate CLOG/PLOG       S    Stop user(s)
  *    Issue reactivate CLOG command       T    Termination Commands
  E    Extended Error Recovery             *    Manage Online Utilities
  *    Force CLOG or PLOG switch           *    User Table Maintenance
  L    Lock or unlock files                *    Replicator Management
  *    Reset ONLINE-DUMP-Status            *    Cluster operator commands
  .    Exit                                ?    Help
 ----  ------------------------------     ----  ------------------------------
       Code ........... _
       Userid(ETID) ... ________
       CLOG/PLOG Ind .. _       Global.. _
       Database ID .... 1955   (WIS1955)             NucID .. 1021



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)

- Extended Error Recovery
- Display Locked Files
- Stop User(s)
- Terminate a Session Normally (ADAEND)

Extended Error Recovery

Selecting option E (extended error recovery) from the Session Opercoms menu displays the Extended Error Recovery menu:

```
14:45:54          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                          -  Extended Error Recovery  -             PACIE02

                    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 .... 1955   (WIS1955)              NUCID .. 1021

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 (adddelete PIN modules) from the Extended Error Recovery menu displays a list of currently available PIN modules:

```
14:47:40          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                -  Add/Delete PIN Modules  -               PACIEM2
NUCID .. 1021

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






PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu        ↩
```

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:

```
14:48:34          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955               -  List/Modify PIN Routines  -              PACIEP2
NUCID .. 1021
Mark entries with 'A' Activate, or 'D' Deactivate:         Total Pins: 267

 M  Condition        Error Location        Status  Uses  Module     Message
 -  --------- ---------------------------- ------- ---- --------  -----------
 _  000C1000  All Locations                Active     0   ADAMXY
 _  000C2000  All Locations                Active     0   ADAMXY
 _  000C3000  All Locations                Active     0   ADAMXY
 _  000C4000  All Locations                Active     0   ADAMXY
 _  000C5000  All Locations                Active     0   ADAMXY
 _  000C6000  All Locations                Active     0   ADAMXY
 _  000C7000  All Locations                Active     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 indicates 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:

```
14:49:15          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                          -  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 .. 55
    UTI/EXF Ind .. U
    Database ID .. 1955   (WIS1955)

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu        ↩
```

Option D from this menu displays the Display Locked Files screen:

```
14:49:15          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
DBID 1955                 -  Display Locked Files  -                PACID02

 Mark entries with 'U' to unlock:
 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:

```
14:50:30          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                               -  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 ...... 55___
    Last Activity .... ________ (elapsed time in seconds)
    Job Name ......... ________
    Purge UQE(s) ..... N
    Selected Userid ..
    Database ID ...... 1955     (WIS1955)             NUCID .. 1021

Command ==>
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----   ↩
```

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).

```
14:51:06          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                           -  Session Termination  -                PACT002

                    Code    Service
                    ----    ------------------------------------
                     A      Normal session termination (ADAEND)
                     *      Cancel session immediately (CANCEL)
                     *      Stop session               (HALT)
                     ?      Help
                     .      Exit
                    ----    ------------------------------------
      Code ......... _
      Database ID .. 1955   (WIS1955)            NUCID .. 1021
                                                 Global.. _

       Current nr. of users in User Queue ... 1
       Nr. of users with open transactions .. 0
       Nr. of active nucs in Plex cluster ... 1


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.

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.

```
14:51:51          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
                             -  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
                   *      Display used storage (DSPACE)
                   ?      Help
                   .      Exit
                  ----    ------------------------------------
    Code ......... _
    File No ...... 55____ 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:

Option	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.

- Display Files
- Display General Database Layout

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:

```
 14:53:05          ***** A D A B A S  BASIC  SERVICES *****          2014-04-30
 DBID 1955                      - Display Files -                    PDRF032

Fnr   File Name        Loaded   Top-ISN    Max-ISN    Extents     Ind.    %Used
                       YY-MM-DD                       NI UI AC DA ACISEXU  A  D
----- ---------------- -------- ---------- ---------- -- -- -- -- ------- -----
    1 EMPLOYEES        09-02-17       1107       1695  1  1  1  1 NNISNNN 77 92
    2 SECURITY         12-09-12          2       2543  1  1  1  1 NNNSNNN 12  5
    3 MISCELLANEOUS    09-02-17       1779       2543  1  1  1  1 NNISNNN 86 53
    6 EXPANDED         09-02-17       1107       1600  1  1  1  1 NNISNXN 74 46
    7 EXPANDED         09-02-17       3107       3600  1  1  1  1 NNISNXN 74 46
    9 TEST-9           10-08-06          0       1695  1  1  1  1 NNNSNNN  0  0
   10 TRIGGER-FILE     09-02-17          1       1695  1  1  1  1 NNISNNN 23 10
   11 NAT-SYSTEM       09-02-17      99317     100063  1  1  1  1 NNISNNU 52 74
   12 NAT-USER         09-02-17        940      30527  1  1  1  1 NNISNNN 80 43
   13 NAT-FDIC         09-02-17         47       5087  1  1  1  1 NNISNNN 33 14
   14 SPAN-TEST-14     13-08-30          1       1695  1  1  1  1 NNNSNNN 55  5
   19 CHECKPOINT       10-12-14      25281      26287  1  1  1  1 NNNSNNN  2 99
   20 FILE-1955-20     09-03-04         16       1695  1  1  1  1 NNNSNNN  0 20
   21 FILE-1955-21     09-03-04          7       1695  1  1  1  1 NNNSNNN  0 10

 PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
 Help     Repos     Exit      Extents   --       -        +        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.
- 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:

```
14:53:56          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
DBID 1955                   - Display File Layout -                 PDRF043
 ****************
 *  File 11     *   NAT-SYSTEM
 ****************

 Records loaded ..... 88764         Date loaded .......... 2009-02-17 19:17:04
 TOP ISN ............ 99317         Date of last update .. 2014-04-28 16:33:27
 Max ISN expected ... 100063        Max Compr Rec Lngth .. 5060
 Minimum ISN ........ 1             Asso/Data Padding .... 10%/10%
 Size of ISN ........ 3 Bytes       Highest Index Level .. 3
 Number of Updates .. 980806        RPLUPDATEONLY. No
 ISN Reusage ........ Yes           USERISN ...... Yes  PGMREFRESH ..... No
 Space Reusage ...... Yes           MIXDSDEV ..... No   NOACEXTENSION .. No
 ADAM File .......... No            Spanned rec .. No   MU/PE indices .. 1
 Ciphered File ...... No            Replication .. No   Privileged Use . No
 Coupled Files ...... None
 Blk per DS Extent .. 0             Logged DSF Changes ... DS AC Index
 Blk per UI Extent .. 0             Total Changed Blks ... 33976
 Blk per NI Extent .. 0             Multi Client File .... 0
                                    Press Enter to display more information
PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit      Refresh                              Menu       ↩
```

```
14:54:40          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
DBID 1955                   - Display File Layout -                 PDRF043
 ****************                                                   Page 2
 *  File 11     *   NAT-SYSTEM
 ****************

 Last format AC ISN .. 100063
 File readonly mode .. No           Date FDT modified .... 1949-12-31 23:59:59
 FDT deleted field ... No
 File has l/c fields . No
 FDT has F,8 field ... No
 FDT w/datetime mask . No
 FDT w/system fields . No           SYFMAXUV value ....... 0

 Free space available for file extents: At least 134  Extents





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:

```
 14:55:02          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
 DBID 1955                  - Display File Layout -                  PDRF052
 File 11

    I Dev  LiI  Space allocated  I      From         To    I Unused
    I Type TyI  Blocks   / Cyls. I      RABN         RABN  I BLOCKS  /  Cyls.
----I--------I-------------------I ------------------------I------------------
    I        I                   I                         I
ASSOI 3390 ACI      118        0 I       5493 -       5610 I        0        0
ASSOI 3390 UII       50        0 I       7313 -       7362 I       35        0
ASSOI 3390 NII     1702        6 I       5611 -       7312 I      814        3
    I        I                   I                         I
DATAI 3390 DSI    44782      298 I        382 -      45163 I    11827       78









 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:

```
14:55:29          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
DBID 1955                - Display General DB-Layout -              PDRG012


 Database Name ............ WIS1955
 Database Number .......... 1955
 Database Version ......... 8.3
 Database Load Date ....... 2010-12-14 11:29:33

 System Files ............. 19 , 2 , 10 , 0 , 0 , 0 , 0
 Maximum Number of Files .. 1000
 Number of Files Loaded ... 33
 Highest File Loaded ...... 400
 Trigger File Number ...... 10

 Size of RABN ............. 3 Bytes
 Current Log Tape Number .. 429
 Delta Save Facility ...... Inactive     Replication Facility ..... Yes
 Recovery Aid Facility .... Inactive
 Universal Encoding Sup. .. Inactive

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 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 General DB-Layout screen lists the current code values:

```
 14:55:54         *****  A D A B A S   BASIC  SERVICES  *****       2014-04-30
   DBID 1955               - 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:

```
14:55:54          ***** A D A B A S  BASIC  SERVICES *****       2014-04-30
DBID 1955                - Display General DB-Layout -              PDRG012

      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 3390 I      53982       200 I          1        53982  I DDASSOR1 I
      I      I                      I                           I          I
 DATA I 3390 I      44990       300 I          1        44990  I DDDATAR1 I
 DATA I 3390 I      44999       300 I      44991        89989  I DDDATAR2 I
      I      I                      I                           I          I
 WORK I 3390 I       8091        60 I          1         8091  I DDWORKR1 I









PF1----- PF2------ PF3------ PF4------ PF6----- PF7----- PF8----- PF12-----
Help               Exit                                          Menu       ↩
```

Index

A

Adabas Cluster Services
  monitor a session online, 241
Adabas Online System
  demo version, 235-236
    error messages for, 240
    invoking, 238
    logging on, 238
    main menu functions, 238
    obtaining help information, 240
    overview, 239
    range of options, 237
    selecting a menu option, 240
    specifying a database for, 239
    using program function (PF) keys, 239
Adabas Parallel Services
  monitor a session online, 241
Adabas SMF records, 197
ADADBS utility
  allocate space using, 143
ADAINV utility
  allocate space using, 145
ADALOD utility
  allocate space using, 146
ADAM
  direct record access without inverted lists, 55
ADAMXY module, 172
ADAORD utility
  allocate space using, 149
ADARAI utility
  concept of generation, 70
  general description, 69
  starting, 70
ADARUN
  display parameters, 242
ADASAV utility
  allocate space using, 149
ADASMXIT, 181
address converter
  space allocation by nucleus, 142
  space calculation using formula, 125
ASMFREC mapping macro, 217
Associator
  automatic space calculation, 119
  blocks reserved by Adabas, 119
  device allocation, 134
  space calculation using formula for address converter, 125
  space calculation using formula for normal index (NI), 120
  space calculation using formula for upper index (UI), 124
ASStI subtype, 199
ASStP subtype, 200
ASStT subtype, 199
autobackout
  function of routine, 66

C

checkpoints
  list current, 252
  types of, 253
ciphering
  data, 62
Command log
  device allocation, 134
commands
  checkpoint, 67
  CL (close), 66
  ET (end transaction), 66
condition description block (CDB), 182

D

data access
  strategies for, 50
data compression
  default options, 56-57
  fixed-storage (FI) options, 56-57
  null-value options, 57
  null-value suppression options, 56
  padding factors, 59
data duplication
  logical, 49
  physical, 49
  types in database design, 49
data protection area
  space calculation using formula, 128
Data Storage
  automatic space calculation, 119
  device allocation, 134
  space allocation by nucleus, 142
  space calculation using formula, 126
database
  define, 115
  define parameters for, 136
  design, 42
  display general layout, 270
  estimate size of, 116
  file and record design, 44

online maintenance menu, 257
online report, 265
recovery and restart design, 63
space management, 137
standards, 16
using status report to control space use, 150
date-time edit masks
conversions between date-time representations, 85
default and null value handling, 90
E(DATE), 80
E(DATETIME), 81
E(NATDATE), 82
E(NATTIME), 83
E(TIME), 80
E(TIMESTAMP), 81
E(UNIXTIME), 84
E(XTIMESTAMP), 84
reference, 79
value verification, 91
DBA
advising on data collection and validation, 14
advising on system development, 14
assisting in database design, 11
database planning, 30
defining database contents, 15
describing data sources, 20
determining responsibility for data, 14
documenting applications using the database, 19
documenting backup procedures, 22
documenting data access and manipulation procedures, 21
documenting database standards, 16
documenting restart and recovery procedures, 23
documenting the database, 16
educating users, 11
establishing database procedures and standards, 9
liaison with user, 28
maintaining procedures and standards, 11
maintaining user ID and password information, 21
management support for, 8
measuring database performance, 23
necessary attributes, 7
position in the IS organization, 7
relationship to operations, 35
selecting applications, 14
selecting suitable applications, 12
summary of functions performed by, 12
training responsibilities, 24
using a data dictionary, 18
working with Software GmbH, 37
descriptor
efficient use of, 51
prefixed with multiclient owner IDs, 74
using ISNs as, 54
device type
description of, 138
device types
allocating Adabas components to, 134

E

E(DATE) edit mask, 80
E(DATETIME) edit mask, 81
E(NATDATE) edit mask, 82
E(NATTIME) edit mask, 83
E(TIME) edit mask, 80
E(TIMESTAMP) edit mask, 81
E(UNIXTIME) edit mask, 84
E(XTIMESTAMP) edit mask, 84
edit masks
date-time, 79
E(DATE), 80
E(DATETIME), 81
E(NATDATE), 82
E(NATTIME), 83
E(TIME), 80
E(TIMESTAMP), 81
E(UNIXTIME), 84
E(XTIMESTAMP), 84
error handling facility, 169
errors
Extended Error Recovery menu, 260
planning for recovery, 64
system/transaction failure, 66
transaction recovery, 65
ETID
conversion to owner ID by ADALOD, 77
exclusive file control, 67
expanded files
and the nucleus, 97
and utilities, 98
define using ADALOD, 94
deleting, 97
description of, 93
general description, 48
inserting a component file, 96
inspecting, 97
recommended nucleus changes for, 98
removing a component file, 96
restrictions when using, 98
rules for defining, 95
extents
description of logical, 140
description of physical, 138

F

Field Definition Table (FDT)
define for a new file
using AOS demo version, 256
fields
advantages and disadvantages of combining, 50
fixed-storage (FI) option, 50, 57
null-value suppression (NU) option, 57
using groups of, 50
using numeric, 50
file coupling
logical, 54
multiclient support for logical files, 72
physical, 53
files
database allocation of large, 135
design of records and, 44
display a list of, 266
display file layout, 268
display locked, 263
exclusive control of, 67
linking multiple physical in a single logical, 48
online maintenance menu, 255

free space table (FST)
  description of, 141

H

hold queue
  display, 246
hyperdescriptor
  description of, 53

I

IBM type 89 SMF records, 219
improved data recording capability
  use recommendation, 136
index
  space allocation by nucleus, 142
ISNs
  user-assigned, 54
  using as descriptors, 54

L

large files
  database allocation with, 135
logical data duplication, 49

M

maintenance
  display levels, 251
MAXISN
  ADALOD parameter, 142
multiclient files
  ADACMP utility support for, 78
  ADALOD utility support for, 76
  ADAULD utility support for, 77
  concept of owner, 71
  concept of superuser, 72
  converting from single client, 70
  data and index structures, 73
  description of, 70
  owner ID stored in user profile, 75
  performance considerations, 75
  relationship of owner ID to user ID (ETID), 71
  response codes, 75
  support for soft coupling, 72
  transparent to application programs, 72
multiple-value fields
  description of, 44
multiprocessing
  monitoring sessions, 241

N

normal index
  space calculation using formula, 120
nucleus
  required components, 116
  space allocation by, 142
  space allocation rules, 142

P

password
  security, 61
performance
  methodology for achieving satisfactory, 42
periodic groups
  description of, 44
phonetic descriptor
  description, 52
physical data duplication, 49
PIN modules
  ADAMXY, 172
  add / delete online, 261
  installing, 173
  list of, 173
PIN routine
  processing, 171
  user exit, 181
PIN routines
  activate / deactivate online, 262
  defined, 171
  display, 262
PINAUTOR, 174
  activating, 173
  installing, 173
  user exit with, 183
PINAVI, 175
PINCOR, 176
PINOPRSP, 176
  activating, 173
  installing, 173
PINRSP, 177
  user exit with, 183
PINSAF, 178
PINUES, 179
Predict
  Adabas data dictionary, 19
Protection log
  device allocation, 134

R

RABNs
  description of, 139
records
  combining multiple types in one file, 47
  design of, 49
  designing files and, 44
  direct access to, 55
recovery
  planning and design, 63
Recovery log
  defining, 70
  device allocation, 134
  general description, 69
resources
  display usage statistics, 247
restart
  planning and design, 63
rules
  nucleus space allocation, 142

S

security
- by ciphering data, 62
- by value, 62
- password and value-level, 61
- planning and facilities, 60
- using ADASAF, 63
- using AOS Security, 63
- using Natural Security, 63

session
- display status, 248
- monitor online, 240-241
- terminate online, 265

SMF records, 197
- Adabas command activity section, 205
- Adabas global cache activity by Adabas file number section, 206
- Adabas global cache activity by block type section, 205
- Adabas global cache activity section, 207
- Adabas global lock activity section, 208
- Adabas internucleus messaging control block activity section, 209
- Adabas internucleus messaging counts section, 209
- Adabas internucleus messaging service time histogram section, 210
- Adabas Parallel Services cache activity section, 211
- Adabsa file activity section, 205
- ADARUN parameter value section, 212
- ASMFREC mapping macro, 217
- header section, 202
- I/O by DD name section, 213
- IBM type 89, 219
- product ID section, 204
- record sections, 200
- record size limites, 198
- record structure, 198
- record subtypes, 199
- self-defining section, 203
- session statistics section, 214
- SMF user exit, 218
- statistics recording, 200
- storage pool section, 215
- subtype 1, Adabas nucleus initialization, 199
- subtype 2, Adabas nucleus termination, 199
- subtype 3, Adabas interval statistics, 199
- subtype 4, Adabas parameter change, 200
- thread activity section, 216
- zIIP statistics section, 216

SMF user exit, 218

Sort
- device allocation, 134

Sort data set
- space calculation using formula, 131

space
- allocate/deallocate procedures, 141
- allocating to Adabas components, 134
- allocation by nucleus, 142
- allocation rules, 142
- allocation using ADADBS, 143
- allocation using ADAINV, 145
- allocation using ADALOD, 146
- allocation using ADAORD, 149
- allocation using ADASAV, 149
- database management, 137
- efficient use of, 56
- estimating for database as a whole, 119
- estimating for each file, 119
- format using ADAFRM utility, 136
- formulas for estimating, 120
- full physical extents, 151
- general procedure for estimating, 119
- maximum logical extents reached, 152
- maximum physical extents reached, 151
- problems and recommended actions, 150

subdescriptor
- description, 52

superdescriptor
- description, 52

system
- display status, 248
- failure, 66

T

Temp
- device allocation, 134

threads
- display usage statistics, 250

transaction
- failure, 66
- recovery, 65
- recovery limits, 67

U

Universal Encoding Support (UES), 185

upper index
- space calculation using formula, 124

user
- stop operation for
  - using Basic Services, 264

user data
- for restart purposes, 68
- reading using direct call command, 66

user exits
- ADASMXIT, 181
- handling errors in, 170
- PIN routine, 181

users
- stop online, 264

W

Work
- data set allocation, 127
- part 1 space calculation using formula, 128
- part 2 space calculation using formula, 129
- part 3 space calculation, 130

Work area
- device allocation, 134
- part 4 space calculation, 130