Adabas

Command Reference

Version 8.5.3

April 2020

ADABAS & NATURAL

This document applies to Adabas Version 8.5.3 and all subsequent releases.

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

Document ID: ADAMF-COMREF-853-20230316

Table of Contents

Command Reference

This documentation describes the powerful and flexible set of Adabas direct call commands for performing database operations. These call commands provide a direct interface to the Adabas database when Natural or another fourth-generation database language is not being used.

Note: Data set names starting with DD are referred to in Adabas documentation with a slash separating the DD from the remainder of the data set name to accommodate z/VSE data set names that do not contain the DD prefix. The slash is not part of the data set name.

The *Adabas Command Reference Guide* documentation is organized in the following parts:

Overview	Provides an overview of Adabas commands and some general programming considerations when making Adabas calls.
Calling Adabas	Describes the calling procedures for Adabas commands for both the ACB and ACBX interfaces.
Adabas Control Block Structures	Describes the Adabas control block (ACB and ACBX) structures you can use when making calls to Adabas.
Adabas Buffer Descriptions (ABDs)	Describes the structure and use of Adabas buffer descriptions (ABDs) when making ACBX interface calls to Adabas. ABDs are only supported in Adabas 8 and later releases.
Defining Buffers	Describes the creation and use of buffer definitions when making calls to Adabas.
Commands	Provides a detailed description of each Adabas command you can use in an Adabas call.
Programming Examples	Provides programming examples of Adabas calls in a variety of host languages.

1 Syntax Conventions

This document describes the syntax conventions used in this documentation for direct calls, buffer descriptions, and buffer specifications. Notation specific to a particular buffer is introduced in the discussion of that area later in this section.

Convention	Identifies...	Description	Example
uppercase, bold	an Adabas keyword	Syntax elements appearing in uppercase and bold font are Adabas keywords. When specified, these keywords must be entered exactly as shown.	`field NC` The syntax element NC is an Adabas keyword.
lowercase, italic, normal font	a variable	Syntax elements appearing in lowercase and normal, italic font identify items that you must supply.	`field i[-j]` The syntax elements *field*, *i*, and *j* are variables. They identify a value you must supply.
vertical bars (|)		Vertical bars are used to separate mutually exclusive choices. **Note:** In more complex syntax involving the use of large brackets or braces, mutually exclusive choices are stacked instead.	`a | b` In the example above, you must select between "a" or "b". There are no defaults.
brackets ([ ])	optional elements or choices	Brackets are used to identify optional elements. When multiple elements are stacked or separated by vertical bars within brackets, only one of the elements may be supplied.	`field [, length]` In this example, the length parameter is optional.
braces ({ })	required elements or choices	Braces are used to identify required elements. When multiple elements are stacked or separated by vertical bars within braces, one and only one of the elements must be supplied.	`{C | S}` In this example, either "C" or "S" must be specified.

Convention	Identifies...	Description	Example
indentation	subparameters	Indentation is used to identify subparameters of a parameter.	

ellipsis (...)	repeated elements	Ellipses are used to identify elements that can be repeated. If the term preceding the ellipsis is an expression enclosed in square brackets or braces, the ellipsis applies to the entire bracketed expression.	`[FIELD='field-name ↩`
			`[, option]...']...`
			In this example, the FIELD parameter can be repeated. In addition, more than one option can be associated with a field.

b...	blank	The italicized, lowercase letter *b*, when used singly or in groups (such as *bbbbbbbb*) indicates a blank or a series of blanks. Each *b* represents one blank.	If the value "ISN*bbbbb*" is specified in this field, it indicates that the ISN values are to be used as the sorting sequence (*bbbbb* represent blanks).
other punctuation and symbols	required punctuation	All other punctuation and symbols must be entered exactly as shown.	`fmtsel redfmt.` In this example, the period is required.

2 About this Documentation

Document Conventions

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

Online Information and Support

Product Documentation

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

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

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

Tech Community

You can collaborate with Software AG 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 AG 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 AG resources.

Product Support

Support for Software AG 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.

I Overview

This part of the Adabas command reference documentation provides a high-level description of the Adabas commands as well as some general programming considerations you should consider when making an Adabas call. The procedures used to make an Adabas call are described in ***Calling Adabas***, elsewhere in this chapter.

The information is organized under the following headings:

3 About Adabas Commands

This section provides an overview of the Adabas commands categorized by function: database query, database modification, Data Storage read, Associator read, logical transaction processing, and special commands.

In addition, Adabas command facilities related to data protection, recovery, and user restart are described. The *transaction* concept is introduced and ET logic operations are explained. Competitive updating is discussed for ET logic (record hold/release and avoidance of resource deadlock) and exclusive control users; nonactivity timeouts are described for all user types.

Command Types

The commands can be categorized into the following functions:

- Database Query Commands (Sx)
- Data Storage Read Commands (L1-L6)
- Associator Read Commands (L9, LF)
- Database Modification Commands (A1, E1, N1/N2)
- Logical Transaction Control Commands (ET/BT)
- Special Commands

Database Query Commands (Sx)

Database query commands (S1/S4, S2, S5) search for and return the ISNs of specified records or record groups according to specified search criteria. Other commands in this category (S8, S9) sort the resulting ISN lists in preparation for later operations.

The ISN lists resulting from any S*x* command may be saved on the Adabas Work data set for later retrieval during your user session.

In most cases, these commands do not actually read the database; ISNs are read directly from the Associator's inverted lists. Options allow the ISN's record to be placed in hold status to prevent its being updated by other programs until the record is released; if desired, additional field values contained in the first ISN's record can be read from Data Storage.

Note: The behavior of nondescriptor searches in Adabas databases differs between mainframe and open systems in regards to null suppression in the fields. In open systems, nondescriptor searches do not return records with null values in a field if the field is null-suppressed (NU); on mainframe systems, the null-suppression (NU) of fields is ignored during nondescriptor searches. At this time, to resolve this problem, we recommend that you remove the null suppression option (NU) for open systems fields, if the fields must be used for a nondescriptor search.

This section covers the following database query commands:

S1/S4

The S1/S4 command selects the records that satisfy given search criteria.

- If only descriptors are used, the query is resolved using the inverted list alone without accessing Data Storage.
- If no descriptors are included in the search criteria, the S1/4 command reads each record in Data Storage to resolve the query.
- If both descriptors and nondescriptors are used within the search criteria, Adabas first searches the inverted list for the descriptors and then reads the Data Storage for all matching ISNs to check the nondescriptors.

S1 and S4 commands return the count of records that satisfy the search criterion, and a list of the ISNs for those records. An option permits the record identified by the first ISN in the resulting ISN list to be read from Data Storage.

The S4 command may be used to place the record identified by the first ISN in the ISN list in hold status. This prevents another user from updating the record until it is released.

S2

The S2 command is equivalent to the S1 command except that the ISNs of the records selected are returned in the sort sequence of a user-specified descriptor (or descriptors). One to three descriptors may be used. Ascending or descending sequence may be specified.

S5

The S5 command is used to select the ISNs of records in one file which are coupled to a record with a given ISN in another file.

You specify the file and ISN for which coupled ISNs are to be returned and the file in which the coupled records are to be selected. Adabas returns the number of records coupled to the ISN and the list of the coupled ISNs.

S8

The S8 command performs logical (AND, OR, or NOT) operations on two ISN lists previously created by an S1/S4, S5, S8, or S9 command.

- AND results in a single ISN list containing ISNs present in both source lists.
- OR results in a single ISN list containing ISNs present in either source list.
- NOT results in a single ISN list containing ISNs present in the first list but not in the second list.

S9

The S9 command sorts an ISN list created previously by a S1/S4, S2, S5, S8 or S9 command.

The ISN list may be sorted by ISN (ascending sequence only), or by one to three user-specified descriptors (ascending or descending sequence).

Data Storage Read Commands (L1-L6)

The L1 through L6 commands are used to read actual records from Data Storage. Depending on the specified command and its options, records are read individually in the sequence in which they are stored, in the order of an ISN list created by one of the database query commands, or in logical sequence according to a user-specified descriptor.

A hold option allows the database records to be locked until released by a separate command or at transaction end.

This section covers the following data storage read commands:

L1/L4

The L1 command reads a single record from Data Storage. You specify the file number, ISN of the record to be read, and the fields for which values are to be returned. Adabas returns the requested field values.

The L4 command is the same as the L1 command except that the record is placed in hold status. This prevents other users from updating the record until it is released.

The multifetch/prefetch option prefetches records on either a session or command basis. This can reduce the overhead for multiple record fetches. The multifetch option is platform-independent.

The GET NEXT option may be used to read one or more records identified by ISNs contained in an ISN list without requiring that you specify each ISN. Usually, the ISN list is created by a previous S*x* command.

The response code option issues a response code 145 (ADARSP145) if the L4 command cannot place a record in hold status because it is being held by another user. Otherwise, you are placed in wait status until the ISN (and record) are freed or the waiting user's transaction is timed out.

The ISN sequence option may be used to read records in ISN sequence. The ISN you specified is read, unless it is not present, in which case the record which has the next higher ISN is read.

L2/L5

The L2 command reads the records from a file in the sequence in which they are physically stored in Data Storage. You specify the file to be read and the fields for which values are to be returned. Adabas returns the requested field values.

The L5 command is the same as the L2 command except that the record read is placed in hold status. This prevents other users from updating the record until it is released. The multi-fetch/prefetch and response code options (see the L1/L4 command description, above) also apply to the L2/L5 commands.

L3/L6

The L3 command reads records from Data Storage in the logical sequence of a user-specified descriptor. You specify the file to be read, the descriptor to be used for sequence control, and the fields for which values are to be returned. Adabas returns the requested field values.

Command Option 2 for the L3/L6 command specifies whether the records are read in ascending or descending order. In addition, Command Option 2 can be used to specify a start value.

The L6 command is the same as the L3 command except that the record read is placed in hold status. This prevents other users from updating the record until it is released.

In addition to the multifetch/prefetch and response code options (see the L1/L4 command description, above), the start value option permits reading to begin at a user-specified value or ISN.

Associator Read Commands (L9, LF)

The L9 and LF commands read information directly from the Associator inverted lists or field definition tables (FDTs), returning either the inverted list values for a specified descriptor or the field definitions for a specified file in the database.

This section covers the following associator read commands:

- L9
- LF

L9

The L9 command returns each value contained in the inverted list for a given descriptor, and the number of records in which the value is contained.

You specify the file and descriptor for which values are to be returned, the value at which the command is to begin, and whether the values are returned in ascending or descending sequence.

LF

The LF command returns the field definitions for a file. You specify the file for which the field definitions are to be returned.

The field definitions for all the fields in the file are returned. Each field definition consists of the field name, level number, standard format, standard length, and definition options.

Database Modification Commands (A1, E1, N1/N2)

Database modification commands (A1, E1, and N1/N2) add, change, or delete database records and update the related Associator lists accordingly. You can assign ISNs to new records or they can be assigned by Adabas.

This section covers the following database modification commands:

- A1
- E1

- N1/N2

A1

The A1 command updates the contents of one or more fields within a record. You specify the file and ISN of the record to be updated, together with the fields to be updated and the values to be used for updating.

Adabas performs all necessary modifications to the Associator and Data Storage. Associator updating is required only if one or more descriptors are updated.

A hold option is available for the purpose of placing the record to be updated in hold status prior to the update.

E1

The E1 command deletes a record or refreshes a file. You specify the file and ISN of the record to be deleted, or specify the file only (without an ISN and command ID) to refresh the file.

Adabas performs all necessary modifications to the Associator and Data Storage.

A hold option is available for the purpose of placing the record to be deleted in hold status prior to the deletion.

N1/N2

The N1/N2 command adds a new record to a file. You specify the file to which the record is to be added together with the fields and field values to be used. Adabas performs all necessary modifications to the Associator and Data Storage.

If the N1 command is used, the ISN for the new record is assigned by Adabas. If N2 is used, you must provide the ISN.

Logical Transaction Control Commands (ET/BT)

An Adabas logical transaction defines the logical start (BT) and end (ET) of the database operation being performed. If the user operation or Adabas itself terminates abnormally, these commands make it possible to restart a user, beginning with the last unsuccessfully processed transaction. ET/BT commands define the transaction start and end, restore pretransaction conditions if a situation occurs that prevents successful completion of the transaction, and read program-specified user data written during the transaction sequence.

Programs that use these commands are called *ET logic programs*. Although not required, Software AG recommends that you use ET logic.

This section covers the following logical transaction control commands:

- BT
- ET

BT

The BT command backs out the current transaction being processed.

All modifications resulting from updates performed during the transaction are removed, and all records placed in hold status during the transaction are released (unless kept in hold status by the multifetch option; see the section ***Multifetch Operation Processing***).

ET

The ET command indicates the end of a logical transaction.

An ET command causes Adabas to physically store all data protection information related to the transaction. This information is used to apply all the updates performed during the transaction at the start of the next Adabas session if the current session is terminated before these updates are physically applied to the database.

The ET command releases all the records that have been placed in hold status during the transaction (unless kept in hold status by the multifetch option; see the section ***Multifetch Operation Processing***). The ET command may also be used to store user data in an Adabas system file. This data may be retrieved with an OP or RE command.

Special Commands

Special commands perform many of the housekeeping functions required for maintaining the Adabas database environment. Commands in this group allow you to perform the following functions:

- Open and close a session (but not to control a transaction).
- Write data protection information and checkpoints.
- Set and release record hold status.

This section covers the following special commands:

- CL
- C1
- C3
- C5
- HI
- OP
- RC
- RE

- RI

CL

The CL command terminates a user session, releasing all records held for that user.

- It physically writes all current data protection information to the data protection log.
- It releases all records currently in hold status for the user.
- It releases all the command IDs and corresponding ISN lists currently assigned the user.
- It stores user data in an Adabas system file (optional).

C1

The C1 command causes a checkpoint to be taken.

The C1 command physically writes all current data protection information to the data protection log, and writes a checkpoint entry to the data protection log and the system checkpoint file. This checkpoint entry may be used as a reference point for subsequent removal or reapplication of updates. An option allows the C1 command to initiate a buffer flush.

C3

The C3 command, issued only by exclusive control and update users (who are not using ET logic), writes a SYNX-03 checkpoint in the Adabas checkpoint file.

- The checkpoint contains the current data protection log and block number.
- The checkpoint may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

If Command Option 2 is specified, the C3 command also stores user data in the Adabas checkpoint file for restart purposes. The stored data may be subsequently read with an OP or RE command.

C5

The C5 command writes user data to the Adabas data protection log.

The data can be read subsequently using the ADASEL utility.

HI

The HI command places a record in hold status. Specify the file and ISN of the record to be placed in hold status.

A record placed in hold status cannot be updated by another user until it is released.

OP

An OP command is mandatory when any of the following apply:

- The nucleus is run with ADARUN parameter OPENRQ=YES
- Exclusive file control (EXF) is to be performed
- User data that was stored in an Adabas system file by a previous ET command is to be read
- User data is to be stored in an Adabas system file, using a C3, CL, or ET command
- You are assigned a special processing priority
- You are an access-only user (no update commands permitted).
- A transaction time limit or a non-activity time limit is to be set for you that differs from that specified by ADARUN parameters TT or TNA*x*, respectively. Your setting must conform to the maximum setting set by the ADARUN parameters MXTT and MXTNA, respectively.
- Special data encoding or architecture is to be specified for your user session.

It may also be used to set the maximum number of records that you can place in hold status at the any given time, and the maximum number of command IDs you may have active at the same time.

RC

The RC command may be used to release one or more command IDs currently assigned to you, or to delete one or all global format IDs.

RE

The RE command reads user data previously stored in an Adabas system file by CL or ET commands.

RI

The RI command releases a record from hold status.

Specify the file and ISN of the record to be released. You may also request that all records currently held by you are released. The RI command should be issued by non-ET users only.

Transactions and ET Logic

This section describes the concept of a *transaction* and an *ET logic user*; it explains ET logic operations including normal and abnormal transaction termination processing and the storage and retrieval of user (ET) data.

This section covers the following topics:

What Is A Logical Transaction?

A *logical transaction* is the smallest unit of work (as defined by the user) that must be performed in its entirety to ensure the logical consistency of the information in the database. Users who use logical transaction commands are referred to as ET logic users.

A logical transaction comprises one or more Adabas commands that read or update the database as required to complete a logical unit of work. A logical transaction begins with the first command that places a record in hold status and ends when an ET, BT, CL, or OP command for the same user is issued.

The RE (read ET data) command can be used to retrieve user restart data stored by the ET or CL command.

Transaction Sequence Number

When a program issues an ET or CL command, Adabas returns a transaction sequence number in the command ID field of the ET or CL command's control block. The transaction sequence number is a count of the total number of ET commands issued thus far during the user session.

The transaction sequence number is set to 1 for the first ET command issued by the user. The first ET command following the OP command returns transaction sequence number 2 in the Command ID field or the Adabas control block. Each subsequent OP command returns the transaction sequence number of the last ET command issued by that same user.

ET Transaction Time Limit

Adabas provides a transaction time limit for programs that use ET logic. The time measurement for a transaction begins when the first command places a record in hold status, and ends when the program issues an ET, BT, or CL command.

The time limit is set with the ADARUN TT parameter; a transaction time limit that overrides this general limit for a specific user can be set with the OP command; this limit is controlled by the ADARUN MXTT parameter.

If a transaction exceeds the prescribed limit, Adabas generates a BT (back out transaction) command. The BT command removes all the updates performed during the transaction and releases all held records.

Adabas returns response code 9 (ADARSP009) for the next command issued by the user indicating that the current transaction has been backed out. The user can either repeat the backed out transaction from the beginning or begin another transaction.

Back Out Transaction (BT) Command

The BT command removes all updates made during the transaction currently being processed. This may be necessary because of a program error, a timeout, or when Adabas determines that the transaction cannot be completed successfully.

A BT command also performs an implied ET command, which releases all the records held during the transaction unless otherwise specified by the multifetch option; for more information, see the section ***Multifetch Operation Processing***.

For example, the command sequence

```
FIND (S4)
  UPDATE (A1) (modify field XX to value 20)
  FIND (S4)
  UPDATE (A1) (modify field YY to value 50)
  END TRANSACTION (ET)
  FIND (S4)
```

```
UPDATE (A1) (modify field XX to value 10)
BACKOUT TRANSACTION (BT)
```

results in the field values of XX = 20 and YY = 50. The second update to field XX is removed by the BT command.

Autobackout

Autobackout, which is performed only for ET logic users, backs out transactions automatically in sessions that end abnormally. Adabas performs an internal BT (back out transaction) followed by autorestart, and then returns response code 9 (ADARSP009) to indicate that the last transaction has been backed out.

Autobackout occurs at the end of any nucleus session that was terminated with HALT.

Autobackout also occurs at the beginning of the next Adabas session to remove any updates that were performed in partially completed transactions by ET logic users during the previous terminated session.

When response code 9 (ADARSP009) indicates that the transaction was backed out, the user has the option of either reissuing the transaction from the start or beginning another transaction.

To restart the backed-out transaction, a terminal operator may need to reenter the data for the transaction, or an internal restart may have to be performed from the beginning of the "update phase" of the transaction; the "update phase" of a transaction begins with the first command that places a record in hold status.

End Transaction (ET) Command

The ET command must be issued at the end of each logical transaction. Successful execution of an ET command ensures that all the updates performed during the transaction will be physically applied to the database regardless of subsequent user or Adabas session interruption.

Updates performed within transactions for which ET commands have not been successfully executed will be backed out by the autobackout routine (see ***Autobackout***).

Unless otherwise specified by the multifetch option, the ET command releases all records held by the user during the transaction. Adabas returns a unique transaction sequence number (see ***Transaction Sequence Number***) that can be used to identify the transaction for auditing or restart purposes.

The ET command may also be used to store user data in an Adabas system file. This data may be used for user restart purposes, and may be read with an OP or RE command.

User Program	Adabas
FIND (S4), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
FIND (S1), HOLD ISN (HI), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
END TRANSACTION (ET)	data protection information for the transaction is written to the Adabas Work and data protection log, ending the transaction
FIND (S4), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
FIND (S1), HOLD ISN (HI), UPDATE (A4)	record updated in Adabas buffer but not necessarily written to the database
. . . Adabas or user session interruption . . .	

When the next Adabas session is started, or when the user is timed out, both updates for transaction 1 are physically written to the database (if they were not previously written). The updates for transaction 2 are not physically written (or will be backed out) because no ET command was processed for this transaction.

User (ET) Data

User data (ET data) may be stored with an ET or CL command and read with an OP or RE (read ET data) command.

One record of user data is kept for each user ID. The data is maintained until the next ET or CL command is issued in which user data is provided.

Each user data record is also written to the Adabas data protection log with each checkpoint written by the transaction. This data may be subsequently read with the ADASEL utility.

User data is primarily used to perform the following functions:

- Store data needed to restart an operation (e.g., input message data, transaction identification data, transaction summary data).
- Store intermediate data to be used by subsequent transactions (e.g., for audit trail purposes).
- Communicate with other users. The data stored may be read by other users provided that the ID of the user who stored the data is supplied in the OP command.

Software AG also suggests using the user data facility to perform the following functions:

- Establish an installation standard for the user data record format and store data for each update transaction with the ET command.
- Read the user data and display it in a standard message format when the user logs on to an application. The user is thus informed of the last successful updating activity which corresponds to his user ID.

Software AG recommends that you keep ET commands that provide user data separate from those that do not. Combining the two types of ET command may cause significant additional overhead by forcing Adabas to repeatedly copy user data from the Adabas Work data set (where it is temporarily stored) to the protection log.

Adabas User ID

The user ID is an identifier used by Adabas to store and retrieve user (ET) data and to assign a special processing priority to a user. The user ID is specified in the Additions 1 field of the OP command.

A user ID provided by the user must meet both of the following criteria:

- It must begin with the character A (X'C1') through 9 (X'F9')
- It must be unique to ensure that the user (ET) data is related to the user regardless of the terminal used. User (ET) data is maintained until the user's next ET or CL command in which user (ET) data is provided.

If the user either does not provide a user ID or provides an invalid user ID, Adabas establishes a default user ID for the user session, and any user (ET) data stored by the user is deleted when the current session is terminated.

To avoid later limitations, Software AG recommends that you always specify a user ID.

Competitive Updating

Competitive updating is in effect when two or more users (in multiuser mode) are updating the same Adabas file(s).

This section describes the Adabas facilities used to ensure data integrity in a competitive updating environment. These include record hold and release processing and the avoidance of resource deadlock, exclusive control updating, and shared hold updating.

This section covers the following topics:

- Exclusive Control Updating

ET User Record Hold and Release

The record hold facility allows the ET user to place a record in hold status for updating without allowing interim updating of the record by another user. Adabas *holds* a record by placing the ISN of the record in the hold queue; as a result, record hold is also called an *ISN hold*.

Record hold is applicable for all ET logic users; Adabas does not place an ISN in hold status for EXU (exclusive control updating) users. Read ***Exclusive Control Updating***, elsewhere in this guide, for more information.

This section covers the following topics:

- Record Hold Commands
- Record Update Using the Hold Option
- Record Release

Record Hold Commands

A record is held when you use the find-with-hold command (S4), the read-with-hold commands (L4, L5, L6), an A1 or E1 command in which the hold option is specified, or a hold record (HI) command. An N1/N2 command issued by an ET logic user also places the added record in hold status.

The successful completion of any of these commands places the record (ISN) in hold status. If the record is already being held by another user, the user issuing the record hold command is placed in a wait status until the record becomes available, at which time Adabas reactivates the command.

If the R (return code) or O (multifetch/return code) option is used with any of the record hold commands and the record to be held is already being held by another user, Adabas returns response code 145 (ADARSP145) instead of placing the user in a wait status.

If you issue a find (S1) or read (L1, L2, L3) command, you can access the record regardless of the fact that the record is in hold status for another user.

Record Update Using the Hold Option

You can update or delete any records they have in hold status by issuing an A1 or E1 command.

An A1 command is executed only if the record is either in hold status for the requesting user, or free from hold status and the you specify that the record should be held. If the record is currently held by another user, your hold request is either placed in wait status or, if you request, you will receive response code 145 (ADARSP145). If the record is not in hold status, response code 144 (ADARSP144) is returned, indicating that the record was placed in hold status.

If an E1 command (without the hold option) is issued for a record that is not in hold status for you, Adabas places the record in hold status for you provided that the record is not in hold status for another user. If you do not place a record to be deleted in hold status, there is no guarantee that the record will not be updated or deleted by another user before the E1 command is executed.

If an N1/N2 command is issued and there is no available space in the hold queue, response code 145 (ADARSP145) is returned.

Record Release

An ET logic user releases records from hold status with the ET command following each logical transaction, and with a close (CL) command at the end of the Adabas session. Programs using ET logic should not release records with the release record (RI) command if any updating has been performed during the current transaction, since this could result in a loss of data integrity.

For example, a record is updated and then released with an RI command by ET user 1, and the transaction continues. In the meantime, the same record is updated by user 2, who then ends the session with an ET command. If user 1's transaction is subsequently backed out due to an error or transaction timeout, the updates performed by both users 1 and 2 are removed even though user 2's transaction completed successfully with an ET command.

Therefore, user programs that employ ET logic should only release records at the completion of a logical transaction with the ET command. Non-ET logic users should use A1 or E1 commands to update or delete records and then release them.

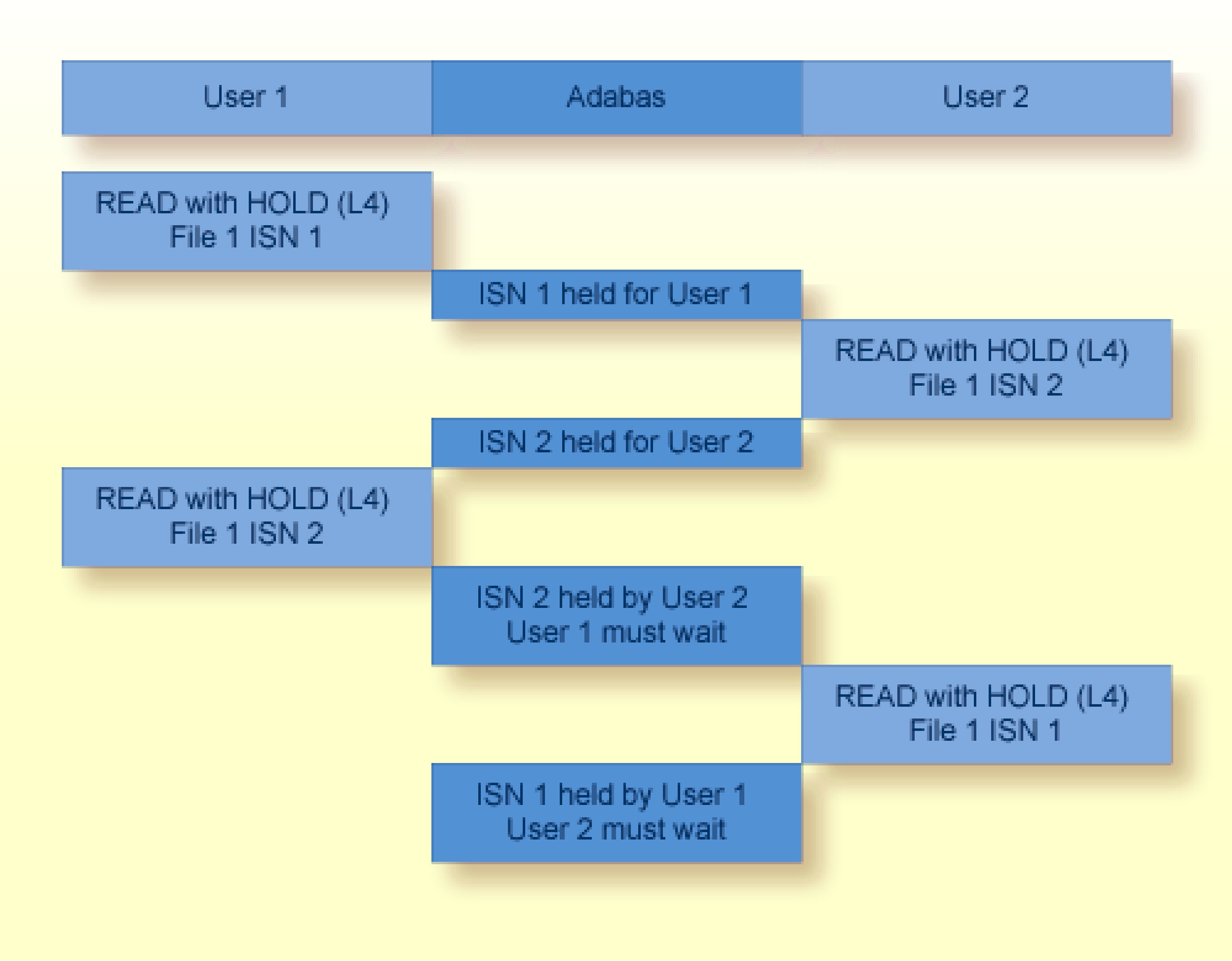
The multifetch option allows records (and their ISNs) to be released from hold status selectively if a non-zero count and the file number/ISN is specified in the ISN buffer when the ET (or BT) command is issued. See the section ***Multifetch Operation Processing***.

The CL command releases all records in hold status for the issuing user, whether an ET user or not.

Avoiding Resource Deadlock

A resource deadlock can occur if two users are placed in wait status because each had requested a record that was currently in hold status for the other user.

For example:


User 1
Adabas
User 2
READ with HOLD (L4) File 1 ISN 1
ISN 1 held for User 1
READ with HOLD (L4) File 1 ISN 2
ISN 2 held for User 2
READ with HOLD (L4) File 1 ISN 2
ISN 2 held by User 2 User 1 must wait
READ with HOLD (L4) File 1 ISN 1
ISN 1 held by User 1 User 2 must wait

Resource Deadlock Example

Adabas protects against such a user deadlock situation by detecting the potential deadlock and returning a response code (ADARSP145) to User 2 after putting the first user in wait status. This occurs if the two users are serviced by the same nucleus (a non-cluster nucleus or the same cluster nucleus). But even in a single nucleus, a deadlock might not be detected if an ISN involved in the deadlock is being held as a shared resource by more than one user. For more information, read about **shared hold status**.

Putting more records in hold status (including shared hold status) may decrease the possible parallelism of transaction processing (and thus, the performance of multiuser applications) and increase the likelihood of deadlocks between transactions (where each transaction holds a record that the other wants).

Shared Hold Status

Shared hold status allows you to lock data records in shared mode, rather than in exclusive mode. This allows your database users to read the same record in parallel transactions, but ensures that no one can update the record concurrently.

Using shared hold status, your users can protect large object values from concurrent updates without locking out other users who may need to read the same LOB value or other LOB values in the same record. It also allows your users to protect the records they read against concurrent updates for specific periods of time:

- For the duration of the read command;
- When the next record in a sequence is read;
- When the user's transaction ends;
- Indefinitely.

Shared hold updating is controlled by four command options for the BT, ET, HI, L4 - L6, RI, and S4 commands, although all four command options do not necessarily apply to all of these commands. These command options are specified in the command option 3 field of ACBX only calls for the associated command. Each command option specifies a different shared hold time period.

- Option C puts the record in shared hold status for the duration of the read operation. It ensures that the version of the record being read has been committed by the last updater. This option is available for the L4, L5, L6 and S4 commands.
- Option Q puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first. It ensures that the record being read cannot be updated concurrently until the next record in the sequence is read (or the transaction is terminated). This option is available for the L4 (when command option 2 is set to "N"), L5, L6 and S4 commands.

  For the S4 command, a command ID must be given and the ISN buffer length must be set to 4. The record returned by the S4 remains in shared hold status until the next record is retrieved by a subsequent S4 or L4/N command with the same command ID.
- Option S puts the record in shared hold status until the end of the transaction. It ensures that the record being read cannot be updated concurrently until the transaction is terminated. This option is available for the HI, L4, L5, L6, RI and S4 commands.

  When specified in an HI request, the record is placed in shared hold status for the user. If a second user also issues an HI request, the record is put in shared hold status for both users.

  When specified in an RI request, if the record is in exclusive hold status and has not been updated in the current transaction, it is placed in shared hold status; if it is in exclusive hold status and has been updated, it is not placed in shared hold status and the exclusive hold remains in effect.

- Option H keeps a record in shared hold status indefinitely (until the next ET or BT command). This option is available only for the BT and ET commands. Records in shared hold status at the time of the BT or ET command are kept in shared hold status beyond the end of the transaction until another ET or BT command is issued (without this H option or the prefetch or multifetch options). Any records in exclusive control are also changed to shared hold status beyond the end of the transaction.

  You cannot use option H if the multifetch or prefetch options are used (or, in Adabas on open systems, if all resources owned by the user are to be released via a command option 1 "T" setting).

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

Putting more records in hold status (including shared hold status) may decrease the possible parallelism of transaction processing (and thus, the performance of multiuser applications) and increase the likelihood of deadlocks between transactions (where each transaction holds a record that the other wants).

Using shared hold status affects the ADARUN NH and NISNHQ parameter settings. Each shared hold request with a different command ID (CID), as well as a (shared or **exclusive**) hold request without a CID, is counted against the NISNHQ and NH limits. This affects application programs that make use of the new Q option for sequential reads. For example, if a program reads records with the Q option and then updates every record, the shared hold operations from the use of the Q option and the exclusive hold operations from the update commands are counted separately. Such a program might need NISNHQ and NH limits set two times larger than when the Q option is not specified.

For more information about these commands and command options, read about the individual commands in ***Commands***, elsewhere in this guide. Detailed description of shared hold status processing and rules are provided in this section.

This section covers the following topics:

- Processing

Duration (Lifetimes) of Shared Hold Status

The following table summarizes how long records remain in shared hold status, depending on which commands are used to place them in and release them from hold status. The key to the codes used in the table are given below the table. Each cell in the table describes the hold status after the command in the top row was used to place the record in hold status and the command in the leftmost column was used to release the record from hold status. If the same record is put in hold status more than once in different ways, it remains in hold status until it has been released from hold status for each way.

Note: In this table, the notation *xx*/*y* indicates the command (*xx*) and the command option (*y*).

Command used to....													
***Release* record from hold status...**	***Place* record in hold status...**												
	BT/H	**ET/H**	**Lx**	**Lx/C**	**Lx/Q**	**Lx/S**	**HI**	**HI/S**	**RI/S**	**S4**	**S4/C**	**S4/S**	**S4/Q**
BT	R	R	R	---	R	R	R	R	R	R	---	R	R
BT/H	S	S	S	---	S	S	S	S	S	S	---	S	S
BT/M	R^{M}	R^{M}	R^{M}	---	R^{M}	R^{M}	R^{M}	R^{M}	R^{M}	R^{M}	---	R^{M}	R^{M}
BT/P[1]	R^{P}	R^{P}	R^{P}	---	R^{P}	R^{P}	R^{P}	R^{P}	R^{P}	R^{P}	---	R^{P}	R^{P}
BT/S[2]	R^{S}	R^{S}	R^{S}	---	R^{S}	R^{S}	R^{S}	R^{S}	R^{S}	R^{S}	---	R^{S}	R^{S}
ET	R	R	R	---	R	R	R	R	R	R	---	R	R
ET/H	S	S	S	---	S	S	S	S	S	S	---	S	S
ET/M	R^{M}	R^{M}	R^{M}	---	R^{M}	R^{M}	R^{M}	R^{M}	R^{M}	R^{M}	---	R^{M}	R^{M}
ET/P[1]	R^{P}	R^{P}	R^{P}	---	R^{P}	R^{P}	R^{P}	R^{P}	R^{P}	R^{P}	---	R^{P}	R^{P}
ET/S[2]	N	N	N	---	N	N	N	N	N	N	---	N	N
Lx/C	S	S	X	S&R	S	S	X	S	S	X	S&R	S	S
Lx/Q	S	S	X	---	R&S	S	X	S	S	X	---	S	R&S
RC	S	S	X	---	R	S	X	S	S	X	---	S	R
RI	R	R	R^{U}	---	R	R	R^{U}	R	R	R^{U}	---	R	R
RI/S	S	S	S^{U}	---	S	S	S^{U}	S	S	S^{U}	---	S	S
S4/C	S	S	X	S&R	S	S	X	S	S	X	S&R	S	S
S4/Q	S	S	X	---	R&S	S	X	S	S	X	---	S	R&S

Key:

Code	Description
---	The record was not in hold status at the end of the first command.
N	The hold status of the record has not been changed.
R	The record has been released from hold status.
R^M	The record has been released from hold status if it was specified in the ISN buffer; otherwise, it is kept in its current (shared or exclusive) hold status.
R^P	The record has been released from hold status if it was *not* specified in the ISN buffer; otherwise, it is kept in its current (shared or exclusive) hold status.
R^S	The record has been released or downgraded to the hold status it had at the time of the specified target savepoint. **Caution:** The record has *not* been put in hold status or upgraded to the hold status it had at the time of the target savepoint.
R^U	The record has been released from hold status; however, if it was updated earlier in the same transaction, it is kept in exclusive hold status.
R&S	The record is released from shared hold status and the next record in the read sequence is placed in shared hold status.
S	The record is in shared hold status.
S^U	The record has been downgraded to shared hold status; however, if it was updated earlier in the same transaction, it is kept in exclusive hold status.
S&R	The record is placed in and released from shared hold status in the course of the same command.
X	The record is in exclusive hold status.

Notes:

1. The prefetch option (P) for ET and BT commands is available only for Adabas on mainframe systems.
2. The savepoint option (S) for ET and BT commands is planned or available in Adabas 6.2 on open systems.

Corollaries:

1. A command that uses the C option has no impact on and is not impacted by the hold status operations of other commands from the same user.
2. A command that uses the Q option, as well as an RC command, may have an impact only on the one record from the read sequence that was previously put in shared hold status using the Q option.
3. An updated record is never released from exclusive hold status before the update has been committed or backed out.

4. An ET or BT command without the H, M, P, or S options always releases all records from hold status, whether they are shared or exclusive.
5. An ET command with the S option has no effect on the hold status conditions of records.
6. A BT command with the S option only releases or downgrades records from hold status. It does not reverse the effects of RI commands. If the application does not use RI commands, a BT command with the S option reinstates the hold status of all records held by the user at the time of the target savepoint.

Rules

The following rules apply to shared hold status and the associated C, Q, S, and H command options:

1. These new command options can only be issued in ACBX calls in the command option 3 field; they are not valid in ACB calls.
2. Only ET users can put records in shared hold status. Access-only (ACC) users and exclusive-update/exclusive-file (EXU/EXF) users who are not also ET users cannot put records in shared hold status. Users with EXU/EXF control on a file cannot run in parallel with users who put records of that file in shared hold status (which must be ET users).
3. Multiple users can have the same record in shared hold status at the same time, whereas only one user at a time can have a record in exclusive hold status.
4. You can, in the course of several commands, put the same record in shared or exclusive hold status more than once. The record is kept in hold status throughout the longest lifetime (duration) of any of the hold status requests.
5. Use of the Q option requires that a command ID be given and, for S4 commands, that the ISN buffer length be set to 4. The Q option is not allowed if the read is in ISN sequence (L4/I) or if the read sequence is performed with multifetch or prefetch (command options M, O, or P).
6. Command options C, Q, and S are not allowed together with the prefetch command option (P).
7. A request to keep held records in shared hold status via the H command option on an ET or BT command applies to all records that were ever put in hold status during the transaction and that have not been released since.

Processing

This section describes the processing behaviors that apply to shared hold status and the associated C, Q, S, and H command options in different situations.

1. A request to put a record in hold status cannot go forward if:
   a. one user has the record in exclusive hold status and another user requests putting it in exclusive hold status, too (existing logic);
   b. one user has the record in exclusive hold status and another user requests putting it in shared hold status; or

   c. one or more users have the record in shared hold status and one of them, or another user, requests putting it in exclusive hold status.

2. If a request to put a record in hold status cannot go forward, the following processing occurs, depending on other command option settings:

   a. If the return option has been specified (command option 1 = "R"), the command is returned immediately with response code 145 (ADARSP145).

   b. If the multifetch return option has been specified (command option 1 = "O"), the records already read and put in hold status by this command are returned. If no record has already been read and put in hold status, the command is returned with response code 145 (ADARSP145).

   c. If the multifetch option has been specified (command option 1 = "M"), the records already read and put in hold status by this command are returned. If no record has already been read and put in hold status, the command is put in wait status until all other users have released the record from hold status, unless putting the command in wait status will cause a deadlock between users (read **point e** in this list for more information about deadlock handling).

   d. If none of these command options has been specified, the command is put in wait status until all other users have released the record from hold status, unless putting the command in wait status will cause a deadlock between users (read **point e** in this list for more information about deadlock handling).

   e. The command is not put in wait status if Adabas detects that this new wait condition would create a deadlock between users who are holding records and waiting to put another one in hold status.

      A deadlock involving multiple users who are holding the same record shared, or (with Adabas Cluster Services or Adabas Parallel Services) who are serviced by different nuclei in a cluster, may not be detected. In this case, the hold commands involved are delayed until one of the waiting users incurs a transaction timeout (TT parameter).

3. While a command is waiting for other users to release a record from shared hold status so that it can put that record in exclusive hold status, further requests by additional users to also put the record in shared hold status will not go forward either.

   A request to upgrade a record from shared to exclusive hold status can go forward in the following scenario:

   - Only one user has the record in shared hold status.
   - Another user is waiting to put the record in exclusive hold status, without first getting it in shared hold status.
   - The first user requests to upgrade the hold status from shared to exclusive. This upgrade request is granted, even though another user is already waiting, because it does not increase the set of users the second user is waiting for.

4. A record in shared hold status can be updated without specifying command option 3 as "H" in the A1 command. If an A1 or E1 command specifies a record that the user has put in shared hold status, the hold status is upgraded to exclusive prior to the update or deletion of the record. (The same applies to an N1 or N2 command if the user has already put the record in shared hold status via a previous HI command.)

   Similarly, a record in shared hold status is upgraded to exclusive if it is read in an L4, L5, L6, or S4 command without the command option 3 settings "S", "Q", and "C", or if it is specified in an HI command without the command option 3 setting of "S".

5. If a request to upgrade a record from shared to exclusive hold status cannot go forward, the following processing occurs, depending on other command option settings:

   a. If the return option has been specified (command option 1 = "R"), the command is returned immediately with response code 145 (ADARSP145).

   b. If the multifetch return option has been specified (command option 1 = "O"), the records already read and put in hold status by this command are returned. If no record has already been read and put in hold status, the command is returned with response code 145 (ADARSP145).

   c. If the multifetch or prefetch option has been specified (command option 1 = "M" or "P"), the records already read and put in hold status by this command are returned. If no record has already been read and put in hold status by this command, the command is put in wait status until all other users have released the record from shared hold status, unless the command cannot be put into wait status because another user is already waiting to upgrade the same record from shared to exclusive hold status (read item **point e** in this list for more information).

   d. If none of these command options has been specified, the command is put in wait status until all other users have released the record from shared hold status, unless the command cannot be put into wait status because another user is already waiting to upgrade the same record from shared to exclusive hold status (read item **point e** in this list for more information)..

   e. The command is not put in wait status if another user is already waiting to upgrade the same record from shared to exclusive hold status. In this case, the new command is returned with response code 145 (ADARSP145) with a new subcode, even though the return option has not been specified.

6. If command option 3 setting "C" is specified for an L4, L5, L6, or S4 command:

   a. The target record is put in shared hold status (unless it was already in hold status for the user prior to the command).

   b. The record is read from the database.

   c. The record is released from shared hold status, unless it had already been in hold status for the user prior to the command with ‘C’ option.

   d. The record read from the database is returned to the user.

This procedure ensures that the returned data will not be the result of an uncommitted update by another user that may still be backed out.

7. If command option 3 setting "Q" is specified for an L4/N, L5, L6, or S4 command:

   a. The target record is put in shared hold status (unless it was already in hold status for the user prior to the command).

   b. The record is read from the database.

   c. The record read from the database is returned to the user.

   d. Unless it is specified in another command that the record be kept in hold status longer, the record is released from shared hold status when one of the following occurs:

      - the next read command in the sequence is issued, or
      - the read sequence is terminated (by releasing the associated command ID).

   e. Furthermore, the record is released from shared hold status when:

      - the user’s transaction ends (e.g., via an ET command) without specifying that the record be kept in hold status (e.g., via the option H), or
      - it is explicitly released from hold status (via an RI command).

   f. If the record is updated, it remains in exclusive hold status until the end of the transaction.

   This procedure ensures that the returned record will not be changed by another user before the user reads the next record in the read sequence or terminates the read sequence or the current transaction.

8. If command option 3 setting "S" is specified for an L4, L5, L6, or S4 command:

   a. The target record is put in shared hold status (unless it was already in hold status for the user prior to the command).

   b. The record is read from the database.

   c. The record read from the database is returned to the user.

   d. The record is released from shared hold status when:

      - the user’s transaction ends (e.g., via an ET command), or
      - it is explicitly released from hold status (via an RI command).

   e. If the record is updated, it remains in exclusive hold status until the end of the transaction.

   This procedure ensures that the returned record will not be changed by another user before the user ends the current transaction. If the user reads the record again directly (via its ISN) prior to transaction end, the record will still exist and be unchanged. If the user attempts to read the record again using the same selection criterion (e.g., in an L4/I, L5, L6, or S4 command), the command might possibly return a different record, which did not satisfy the selection criterion earlier but does satisfy it now.

9. If command option 3 setting S is specified for an HI command:

a. The target record is put in shared hold status (unless it was already in hold status for the user prior to the command).

b. The record is released from shared hold status when:

- the user's transaction ends (e.g., via an ET command), or
- it is explicitly released from hold status (via an RI command).

c. If the record is updated, it remains in exclusive hold status until the end of the transaction.

This procedure ensures that the specified record will not be changed by another user before the user ends the current transaction.

10. By default, all shared and exclusive hold status conditions of records being held by a user are released when the user's current transaction ends. In particular, they are released when the user issues a BT or ET command without command option 1 set to "M", "P" or "O", or command option 3 is set to "H"; or when Adabas decides for internal reasons to back out the transaction.

    For a BT or ET command with multifetch option (command option 1 = "M"), only the shared and exclusive hold status conditions for the records specified in the multifetch buffer are released; all others are kept as is.

    For a BT or ET command with prefetch option (command option 1 = "P"), only the shared and exclusive hold status conditions for the records not specified in the ISN buffer are released. The current (shared or exclusive) hold status conditions for the records specified in the ISN buffer are kept as is. Any additional records specified in the ISN buffer are put in exclusive hold status (with response code 2 (ADARSP002) issued if that is not immediately possible).

11. If command option 3 setting "H" is specified for a BT or ET command, Adabas keeps in shared hold status all records that were put in (or downgraded to) shared hold status earlier in the transaction (or kept in shared hold status at the end of the preceding transaction) and not released since then. Furthermore, Adabas downgrades all records to shared hold status that were put in (or upgraded to) exclusive hold status earlier in the transaction (or kept in exclusive hold status at the end of the preceding transaction) and not released since then.

    When combined with command option 3 setting "Q" for the L4/N, L5, L6 and S4 commands, the "H" option can be used to perform multiple transactions in the course of one read sequence, without losing, at transaction end, the protection of the current record in the sequence against concurrent updates by other users.

    When combined with command option 3 setting "S" for the HI, L4-L6 and S4 commands, the H option can be used to perform multiple transactions in the course of one multi-step, user-defined operation, without losing, at transaction end, the protection of the held records against concurrent updates by other users.

12. If command option 3 setting "S" is specified for an RI command, the target record is downgraded from exclusive to shared hold status, provided it is in exclusive hold status for the user and has not been updated in the current transaction. If the target record is not in exclusive hold status for the user, the RI command has no effect. If the record has been updated in the current

transaction, response code 113 (ADARSP113) is returned (if ISN=0 was specified, response code 2, ADARSP002, is returned).

For an RI command without command option "S", the target record is released from shared or exclusive hold status, provided it is in hold status for the user and has not been updated in the current transaction. If the target record is not in hold status for the user, the RI command has no effect. If the record has been updated in the current transaction, response code 113 (ADARSP113) is returned (if ISN=0 was specified, response code 2, ADARSP002, is returned).

13. If you put the same record in shared or exclusive hold status more than once, the record is kept in hold status throughout the longest lifetime of any of the hold status requests. In particular:

    a. If a record has been updated, it stays in exclusive hold status until the end of the transaction. If the end of the transaction is triggered by a BT or ET command with the "M" (multifetch) or "P" (prefetch) options, you can retain the record in exclusive hold status beyond the end of the transaction. If the end of the transaction is triggered by a BT or ET command with the "H" option, the record is downgraded to and retained in shared hold status beyond the end of the transaction.

    b. If a record has been put in exclusive hold status using an HI, L4-L6 or S4 command and has not been updated, it stays in exclusive hold status until the end of the transaction or until its release via an RI command, whichever occurs first. If the end of the transaction is triggered by a BT or ET command with the "M" (multifetch) or "P" (prefetch) options, you can retain the record in exclusive hold status beyond the end of the transaction. If the end of the transaction is triggered by a BT or ET command with the "H" option, the record is downgraded to and retained in shared hold status beyond the end of the transaction.

    c. If a record has been put in shared hold status using an HI, L4-L6 or S4 command with the "S" option, it stays in shared hold status at least until the end of the transaction or until its release via an RI command, whichever occurs first. If the end of the transaction is triggered by a BT or ET command with the "H" or "M "or "P" options, you can retain the record in shared hold status beyond the end of the transaction.

    d. If a record has been put in shared hold status using an L4/N, L5, L6 or S4 command with the "Q" option, it stays in shared hold status at least until the next read command in the read sequence, the termination of the read sequence (e.g., via an RC command), the end of the transaction, or until its release via an RI command, whichever occurs first.

       If the end of the transaction is triggered by a BT or ET command with "H" or "M" or "P" option, the user can retain the record in shared hold status beyond the end of the transaction.

    e. If a record has been put in shared hold status using an L4-L6 or S4 command with the "C" option, it stays in shared hold status only until the end of the command, unless it had already been in hold status prior to the command.

    A record remains in exclusive hold status until no reason exists anymore to keep it in this status according to items a or b in the list above. A record remains in shared hold status until no reason exists anymore to keep it in this status, according to rules c, d or e.

14. If you end a transaction using a BT or ET command without the "H", "M" or "P" options, all records held by your processing are released from their shared or exclusive hold status at that time.

    If you end a transaction using a BT or ET command with the "H" option, all records that were put in shared hold status earlier (using the "Q" or "S" option) and not released since then are kept in shared hold status for the next transaction. Records that were put in exclusive hold status and not released since then are downgraded to shared hold status.

15. When you put a record in either shared or exclusive hold status, a new transaction starts for you, if one is not is already active. The transaction is subject to the usual transaction timeout logic.

    If you keep records in shared or exclusive hold at the end of a transaction (via command option "H", "M", or "P" on the BT or ET command), a new transaction starts immediately.

    > **Caution:** If an application uses the command option "H" too liberally, one user can prevent a second user from putting a record in exclusive hold status indefinitely, across multiple transactions of the first user, until the transaction of the second user incurs a timeout and is backed out.

    If an RI command releases the only record in (shared or exclusive) hold status for the user, the user's transaction ends. (Note that the record cannot have been updated).

Exclusive Control Updating

You may request exclusive control of one or more Adabas files to prevent other users from updating the file during the execution of your user session. Adabas exclusive file control occurs in one of the following ways:

- **UTI** control is required by long-running, external utilities (such as the ADALOD utility) as well as by short-running utility functions that are executed by the nucleus (for example, using the Adabas Online System or ADADBS functions). It is not available to your application programs. UTI file control is incompatible with any other use of the file; other users may not access or update the file while it is under UTI file control.
- **EXF** control can be specifically requested by your application programs. It is requested when "EXF=*file-number*" is specified in the record buffer of an OP command and it is relinquished by a CL command. EXF file control is incompatible with any other use of the file (except for online save requests); other users may not access or update the file while it is under EXF file control.
- **EXU** control is required by external utilities (such as the ADAULD utility) and can be specifically requested by your application programs. It is requested when "EXU=*file-number*" is specified in the record buffer of an OP command and it is relinquished by a CL command. EXU file control is incompatible with any other update access requests for the file (except for online save requests); other users may access, but not update the file while it is under EXU file control.

When requested, EXF or UTI control is given only for a file if the file is not already in use by another user or utility; EXU control is given only if the file is not already opened for update by another user. If the exclusive control request is denied, Adabas returns response code 48 (ADARSP048).

In addition to preventing competitive updating of a file, exclusive control may be used to simplify recovery procedures, in that the files may be restored regardless of other user activity.

The record hold commands need not (but may) be used for files for which the user has exclusive control. Adabas disables hold logic processing for files being updated under exclusive file control.

Exclusive control users are assumed to be non-ET logic users, and therefore not controlled by the ET timeout restrictions. However, if a non-ET user issues an ET command, that user automatically becomes an ET logic user and is subject to transaction timeout restrictions if records are put in hold status for any reason. That user retains exclusive control until either finished or timed out.

Users performing exclusive control updating can use the C1 command to request that a checkpoint be taken. The C1 checkpoint acts as a reference point to either remove updates that have been applied after the checkpoint, or to reapply updates that were applied before the checkpoint.

Note that a user (application) is accessing a file if the file is listed in the user's User Queue Element (UQE) file list. The file list identifies each file the user (application) is using and the type of file usage (ACC, EXU, EXF, or UPD). A file is added to the UQE file list when:

- it is requested for UTI use by a utility or Adabas Online System;
- it is specified in the record buffer of an OP command following one of the ACC, EXF, or EXU keywords;
- it is specified in the record buffer of an OP command with Command Option 1 set to "R" following the UPD keyword; or
- it is the target of a command without hold (ACC use of the file) or a command with hold (UPD use of the file) by a user who did not start the Adabas session with an OP command specifying Command Option 1 set to "R".

A file is deleted from the UQE file list when:

- a utility or Adabas Online System function with UTI control over the file ends;
- the user session ends; or
- the user's current transaction ends (e.g., in the course of an ET or BT command) and the file had not been added to the file list by the user's OP command.

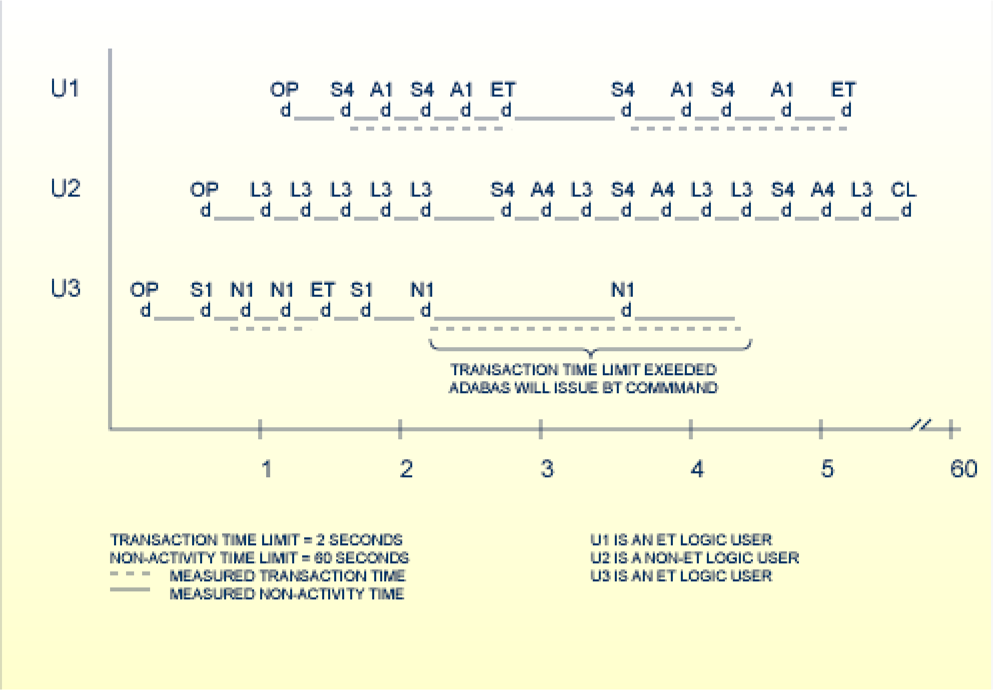
For more information about file usage by utilities and possible resource conflicts, read *Utility Usage of Files and Databases* and *Possible Resource Conflicts*, in the *Adabas Operations Manual*.

Non-Activity Time Limit

All users (access-only, ET logic, or exclusive control) are subject to a non-activity time limit. Different non-activity time limits may be defined for each user type with the ADARUN TNAA, TNAE, and TNAX parameters (for more information, read *Adabas Initialization (ADARUN Statement)* in *Adabas Operations Manual*).

A user-specific non-activity time limit may also be set with the OP command; the maximum is controlled with the ADARUN MXTNA parameter. For more information, read ***OP Command: Open User Session***, elsewhere in this guide.

The following diagram summarizes the actions taken by Adabas when a user exceeds the non-activity time limit.


U1
OP S4 A1 S4 A1 ET S4 A1 S4 A1 ET
d d d d d d d d d d d
U2
OP L3 L3 L3 L3 L3 S4 A4 L3 S4 A4 L3 L3 S4 A4 L3 CL
d d d d d d d d d d d d d d d d d
U3
OP S1 N1 N1 ET S1 N1 N1
d d d d d d d d
TRANSACTION TIME LIMIT EXEEDED
ADABAS WILL ISSUE BT COMMAND
1
2
3
4
5
60
TRANSACTION TIME LIMIT = 2 SECONDS
NON-ACTIVITY TIME LIMIT = 60 SECONDS
MEASURED TRANSACTION TIME
MEASURED NON-ACTIVITY TIME
U1 IS AN ET LOGIC USER
U2 IS A NON-ET LOGIC USER
U3 IS AN ET LOGIC USER

Transaction and Non-Activity Time Limits

This section describes these Adabas actions in the following topics:

- For an ET Logic User
- For an Exclusive File Control User (EXF)

For an ET Logic User

For an ET logic user, Adabas performs the following processing:

- It issues a backout transaction (BT) command for the current user transaction (if necessary).
- It releases all records held during the transaction.
- It deletes the user's file list in the UQE.
- It deletes all command IDs for the user.

Adabas returns response code 9 (ADARSP009) to the user when the next command is issued if one of the following conditions occur:

- The user was not in ET status when the timeout occurred.
- The user was in ET status when the timeout occurred and provided a non-blank user ID in the OP command.

If the user was at ET status when the timeout occurred and did *not* provide a non-blank user ID in the OP command, the user's UQE is deleted.

For an Exclusive File Control User (EXF)

For an exclusive file control user, Adabas deletes the user's file list in the UQE. As a result, the user loses exclusive control of the file or files for which exclusive control was in effect.

In addition, all command IDs for the user are released and the user is changed to an access-only user.

For an Access Only User

For an access-only user, Adabas deletes the user's file list in the UQE.

4 General Programming Considerations

This section explains several concepts that are important to consider when programming calls.

Internal IDs can be specified to perform important functions during Adabas command execution. In the control block of an Adabas direct call command, you can specify a:

- Command ID (ACBCID or ACBXCID). This is a non-blank, non-zero value specified in the ACB or ACBX that acts as an internal ID for the command processing. It can be used to eliminate repeated interpretation and conversion by successive commands that use the same format buffer.
- Format ID. This is a separate four-byte internal ID for decoded format buffers defined either as user-specific or globally available (a *global format ID*) to other users running on the same Adabas nucleus.

The uses of these IDs is explored in detail in this section.

Also described in this section are the procedures used to retrieve ISNs from the Adabas Work and the multifetch and prefetch options. Multifetch and prefetch options are used to reduce execution time for programs that process large amounts of data in sequential order by reducing the number of system commands needed to the complete the Adabas call.

Command, Format, and Global Format IDs

The *command ID*, specified in the Adabas control block (ACBCID or ACBXCID), performs important functions during Adabas command execution. It is an automatically generated or user-specified nonblank, nonzero value that performs the following functions:

- It prevents repetitive format buffer decoding by acting as an internal ID for decoded record formats.
- It tags ISN lists generated by the S*x* command for later access, and saves ISN list overflow.
- It tags (identifies) sequential read processes through sets of records.

If desired, a separate internal *format ID* for decoded format buffers can be specified. This value is identified by a flag in the high-order (leftmost) two bits of the first byte of the Additions 5 field. Depending on the flag value, the format ID can be user-specific (an individual format ID) or available to other users running on the same Adabas nucleus (a global format ID).

This section covers the following topics:

Specifying Command, Format, and Global Format IDs

The following table summarizes the Adabas control block (ACB or ACBX) settings required to specify command IDs, format IDs, and global format IDs.

ID	ACB or ACBX Field Specifications	
	Command ID (ACBCID or ACBXCID)	Additions 5 (ACBADD5 or ACBXADD5)
Command ID	Set this field to any non-blank, non-zero value. Specifying a command ID value of X'FFFFFFFF' causes Adabas to generate command IDs automatically, beginning with X'00000001' and incrementing by 1 for each new command ID.	High-order, leftmost bits set to binary "00".
Format ID	Set this field to any non-blank, non-zero value. Specifying a command ID value of X'FFFFFFFF' causes Adabas to generate command IDs automatically, beginning with X'00000001' and incrementing by 1 for each new command ID.	Set the high-order, leftmost two bits of the first byte to binary "10"; set the fifth through eighth bytes to the format ID. The ID may not start with X'FE' or X'FF'. In non-mainframe Adabas environments, set the first byte of the Additions 5 field to be any lowercase letter.
Global Format ID	Set this field to any non-blank, non-zero value. Specifying a command ID value of X'FFFFFFFF' causes Adabas to generate command IDs automatically, beginning with X'00000001' and incrementing by 1 for each new command ID.	Set the high-order, leftmost two bits of the first byte binary "11"; set the remaining bytes to the ID. All eight bytes of the Additions 5 field are used as a global format ID. In mainframe environments, the first byte may be assigned in the hexadecimal range E2 through E9 (characters S-Z); all other ranges are reserved for use by Software AG. In non-mainframe Adabas environments, the first byte may be assigned in the hexadecimal range 51-5A (characters S-Z).

Notes:

1. When B'00' is set in the high-order (leftmost) two bits of the first byte of the Additions 5 field, the command ID is automatically used as the format ID.

2. When B'10' is set in the high-order (leftmost) two bits of the first byte of the Additions 5 field, separate values are used for the command ID and the format ID, and the fifth through eighth bytes of the Additions 5 field are used as the format ID.
3. When B'11' is set in the high-order (leftmost) two bits of the first byte of the Additions 5 field, separate values are used for the command ID and the format ID, the format ID is treated as a global format ID and all eight bytes of the field are used as the global format ID.
4. Adabas does not verify the assignment of global format IDs. It is your responsibility to ensure that only global format IDs in the allowable range are assigned. Software AG can neither enlarge the range of global format IDs available to users nor make any changes to its products to resolve a global format ID assignment problem.

Command IDs for Read Sequential Commands

The read sequential commands (L2/L3, L5/L6, L9) require that a command ID be specified. The command ID is needed by Adabas to return the records to the user in the proper sequence. These command IDs are maintained by Adabas in the table of sequential commands.

The command ID value provided with these commands is also entered and maintained in the internal format buffer pool unless a separate format ID is provided, as described in the section ***Using Separate Command and Format IDs***. The command ID is released by Adabas when an end-of-file condition is detected during read sequential processing.

Command and Format IDs for Read, Update, and Find Commands

The read commands (L1-L6, L9) and update commands (A1/A4, N1/N2) require a format buffer that identifies the fields to be read or updated. This format buffer must be interpreted and converted into an internal format buffer by Adabas. Using a valid command ID avoids repeated interpretation and conversion by successive commands that use the same format buffer.

A read or update command with a valid command ID causes Adabas to check whether the command ID is in the internal format buffer pool. If the command ID is present, its internal format buffer is used, and no format buffer reinterpretation is required.

Note: When reading or updating a series of records that use the same format buffer, processing time can be significantly reduced if you use a command ID.

Internal format buffers (and the format IDs) resulting from L9 commands can only be used by other L9 commands. Moreover, L9 commands cannot use non-L9 internal format buffers / format IDs. This is also true of global format buffers (and global format IDs) and L9 commands.

When reading and updating the same fields (for example, L5 followed by A1), Software AG also recommends that the same command ID be used for both commands (see the A1/A4 and N1/N2 commands for restrictions on using the same format buffer for reading and updating).

If the read-first-record option is used with an S1/S2/S4 command and a command ID is specified, the command ID and the resulting internal format buffer are also stored in the internal format buffer pool.

If the internal format buffer pool is full and a command is received with a command ID without an internal format in the pool, Adabas overwrites the longest unused entry in the pool with the new interpreted format ID. If a command is subsequently received that uses the deleted command ID, the reinterpreted format buffer for that command ID replaces the next-longest unused entry in the pool. For this reason, programs must not change the format buffer between successive read or update commands with the same command ID. Note, however, that use of a command ID does not guarantee that the format buffer is not reinterpreted.

Using Separate Command ID and Format IDs

It is possible to use separate values for command IDs and format IDs. As long as the high-order (leftmost) two bits of the Additions 5 field are set to binary '00', the command ID is automatically used as the format ID. If, however, the Additions 5 field's high-order two bits are binary '10', the fifth through eighth bytes (Additions 5 + 4(4)) of the field are used as the format ID. Note that the ID may not start with X'FE' or X'FF'.

Note: To identify the format ID as separate from the command ID, non-mainframe Adabas environments expect the first byte of the Additions 5 field to be any lowercase letter. When using separate format IDs in a heterogeneous environment, it is important to identify them alike across all platforms used in the system.

Using a Global Format ID

Particularly in an online environment, multiple users of the same program often read or update the same fields of a file and therefore use identical format buffers.

- When you use the *individual* format ID option, Adabas must store the same internal format buffer for each user.
- When you use the *global* format ID option, a single internal format buffer is shared by many users and the need for Adabas to overwrite internal format buffer pool entries is reduced. This option identifies the format buffer to each user by format ID only, rather than by both format ID and terminal ID. A command ID cannot be designated as a global format ID; in addition, the restriction of L9 formats and their IDs being valid only for use by other L9 commands also applies to global formats and IDs.

The global format ID option is activated by setting the high-order (leftmost) two bits of the first byte of the Additions 5 field to binary '11' (see *Specifying Command, Format, and Global Format IDs*). This causes all eight bytes of Additions 5 to be recognized as the global format ID.

Note: To identify the format ID as global in non-mainframe Adabas environments, the first byte of the Additions 5 field must be set to be any digit or uppercase letter. When using global format IDs in a heterogeneous environment, it is important to identify them alike across all platforms used in the system.

The first byte of the global format ID may be assigned in the hexadecimal range E2-E9; characters S-Z. All other ranges are reserved for use by Software AG.

The allowable range of values for global format IDs in non-mainframe Adabas environments is hexadecimal 51-5A; characters S-Z.

Caution: Adabas does not verify the assignment of global format IDs. It is the user's responsibility to ensure that only global format IDs in the allowable range are assigned. Software AG can neither enlarge the range of global format IDs available to users nor make any changes to its products to resolve a global format ID assignment problem.

A global format ID can be deleted using the RC command and specifying the global format ID in the Additions 5 field, as described.

Command IDs Used with ISN Lists

If a command ID is specified for any command which results in an ISN list (S1,S2,S4,S5,S8,S9), the command ID value may be used to identify the list at a later time.

- If the save-ISN-list option is used for an S*x* command, a command ID must be provided. The save-ISN-list option causes the entire ISN list to be stored on the Adabas Work. ISNs from the list may subsequently be retrieved by an S*x* command or by using the GET NEXT option of the L1/L4 command.
- If the save-ISN-list option is not used and an ISN buffer overflow condition occurs (the entire ISN list cannot be inserted in the ISN buffer), the overflow ISNs will be stored on the Adabas Work only if a command ID value was used. In this case, the command ID and the ISN list it identifies will be released by Adabas when all the ISNs have been returned to the user.

Automatic Command ID Generation

Automatic command ID generation may be invoked by specifying a command ID value of X'FFFFFFFF'. This causes the Adabas nucleus to generate command IDs automatically, beginning with X'00000001' and incrementing by 1 for each new command ID. Automatic command ID generation may not be desirable in all cases; refer to the section ***Command and Format IDs for Read, Update, and Find Commands***.

Releasing Command IDs

You can release a command ID and its associated entries (or ISN list) with an RC command, a CL command, or by using the release-CID option of any S*x* command (S1, S2, S4, S5, S8, S9).

The RC command contains options that allow you to release only those command IDs contained in the internal format buffer pool, the table of sequential commands, or the table of ISN lists.

The CL command causes all the command IDs currently active for you to be released.

The release-CID option of an S*x* command causes the CID specified to be released as the first action taken by the command.

Internal Identification of Command IDs

Each command ID entry is identified by Adabas using an internal user ID together with the command ID value. As a result, one user need not be concerned with the command ID values in use by another. However, a user should avoid using the same command ID value for different commands, particularly if the command ID is used for sequential read (L2/L5, L3/L6, L9) commands and S*x* commands.

Examples of Command ID Use

This section covers the following topics:

- Example 1 : Find / Read Processing
- Example 2 : Find / Read Using the GET NEXT Option
- Example 3 : Read / Update Processing
- Example 4 : Read / Find Processing

Example 1 : Find / Read Processing

A set of records is to be selected and read. The same format buffer is to be used for each record being read.

```
FIND (S1)           CID=EX1A
READ (L1)           CID=EX1B
READ (L1)           CID=EX1B
```

Example 2 : Find / Read Using the GET NEXT Option

A set of records is to be selected and read using the GET NEXT option of the L1/L4 command.

```
FIND (S1)           CID=EX2A
READ (L1)           CID=EX2A
READ (L1)           CID=EX2A
READ (L1)           CID=EX2A
```

Example 3 : Read / Update Processing

A file is to be read and updated in sequential order. The same format buffer is to be used for reading and updating.

```
READ PHYS SEQ (L5)  CID=EX3A
UPDATE (A4)         CID=EX3A
READ PHYS SEQ (L5)  CID=EX3A
UPDATE (A4)         CID=EX3A
```

Example 4 : Read / Find Processing

A file is to be read in logical sequence. A find command is to be issued to a second file using the value of a field read from the first file, and the records that result from the find command are then to be read using the GET NEXT option.

```
READ LOG SEQ (L3)   CID=EX4A
FIND (S1)           CID=EX4B
READ (L1)           CID=EX4B
READ (L1)           CID=EX4B
READ LOG SEQ (L3)   CID=EX4A
FIND (S1)           CID=EX4B
READ (L1)           CID=EX4B
```

ISN List Processing

This section discusses the procedures used to retrieve ISNs from the Adabas Work data set. If the GET NEXT option of the L1/L4 command is used to read the records that correspond to the ISNs contained in the ISN list, ISN handling as discussed in this section is performed automatically by Adabas, and the user need not make use of these procedures.

The notation S*x* command as used in this section refers to any command that may result in an ISN list (S1/S2/S4/S5/S8/S9).

This section covers the following topics:

- Storage of ISN Lists
- Retrieval of ISN Lists

Storage of ISN Lists

Adabas stores ISNs on the Work data set under either of the following conditions:

- An S*x* command is issued, a nonblank nonzero command ID is specified, and the save-ISN-list option is specified. The entire resulting ISN list is stored.
- An S*x* command is issued, a non-blank non-zero command ID is specified, the save-ISN-list option is not specified, and the resulting ISN list contains more ISNs than can be inserted in the ISN buffer. Only the overflow ISNs are stored in this case.

If an S*x* command is issued with blanks or binary zeros in the command ID field, Adabas does not store any ISNs on the Adabas Work.

Retrieval of ISN Lists

You can retrieve ISNs stored on the Adabas Work data set by issuing an S*x* command in which the same command ID value is used as was used for the initial S*x* command. When an S*x* command with an active command ID value is issued, Adabas uses this as an indicator that you are requesting ISNs from an existing ISN list. Adabas locates the ISN list identified by the specified command ID and inserts the next group of ISNs in the ISN buffer. As many ISNs are returned as can fit in the ISN buffer.

This section covers the following topics:

Save-ISN-List Option Specified

If the save-ISN-list option was specified with the S*x* command used to create the ISN list, Adabas uses the ISN specified in the ISN lower limit field to determine the next group of ISNs to be returned.

The next group begins with the first ISN that is greater than the ISN specified in ISN lower limit.

- If binary zeros are specified, the next group begins with the first ISN in the list.
- If a value is specified which is greater than any ISN in the list, response code 25 (ADARSP025) is returned.

If the ISN list was created using an S2 command, the ISN specified must be present in the ISN list. Use of the save-ISN-list option thus permits the user to skip forward and backward within an ISN list. This is useful for programs that must perform forward and backward screen paging.

Save-ISN-List Option Not Specified

If the save-ISN-list option was not specified with the S*x* command used to create the ISN list, Adabas returns the ISNs in the order in which they are positioned in the list, and deletes each group from the Work when it has been inserted in the user's ISN buffer. The command ID used to identify the list is released when the last group of ISNs has been returned to the user. The ISN lower limit field is not used in this case, unless processing is to begin above a specified ISN range.

Using the ISN Quantity Field of the Control Block

The user can determine when all of the ISNs in a list have been retrieved using the ISN quantity field of the control block.

- The first S*x* command returns the total number of records that satisfy the search criteria in this field.
- In addition, each subsequent S*x* command used to retrieve ISNs from the Adabas Work uses this field to store the number of ISNs that were inserted in the ISN buffer.

ISN List Processing Examples

This section covers the following topics:

- Example 1 : Using Sx Command with L1/L4 Commands with GET NEXT Option
- Example 2 : Using the Save-ISN-List Option
- Example 3 : With ISN Overflow Handling
- Example 4 : Without ISN Overflow Handling

Example 1 : Using Sx Command with L1/L4 Commands with GET NEXT Option

```
Sx command
L1/L4 command with 'GET NEXT' option
L1/L4 command with 'GET NEXT' option
L1/L4 command with 'GET NEXT' option
```

The following examples show various results according to the size of the ISN buffer. Positioning for ISN list is defined by ISN lower limit.

1.
   ```
   ISN Buffer Length = 0
   read the ISN list to the work area
   ```

2.
   ```
   L1/L4 with GET NEXT
   result: first ISN's record ↩
   ```

1.
   ```
   ISN Buffer Length = 4
   read first ISN with S1
   ```

2.
   ```
   L1/L4 with GET NEXT
   result: second ISN's record
   ```

1.
   ```
   ISN Buffer Length = 12
   read first ISN with S1
   first 3 ISNs are returned in ISN buffer
   ```

2.
   ```
   L1/L4 with GET NEXT
   result: read fourth/fifth/sixth ISNs' records
   ```

Example 2 : Using the Save-ISN-List Option

Initial S*x* call using save-ISN-list option:

```
Command = Sx
Command ID = SX01              (save-ISN-list option)
Command Option 1 = H
ISN Lower Limit = 0
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total matching ISNs in stored list)

Resulting ISN list: (all ISNs are stored on Work):

```
8    12  14  15  24  31  33
```

Resulting ISN buffer:

```
8    12  14  15  24
```

Subsequent S*x* call:

```
Command = Sx
Command ID = SX01
ISN Lower Limit = 24     (limit ISN choice to 24, +)
ISN Buffer Length = 20   (space for 5 ISNs from ISN list)
CALL ADABAS ...
```

Resulting ISN quantity = 2 (total ISNs returned in ISN buffer)

Resulting ISN buffer:

```
31   33   14  15  24.... remainder of ISN buffer unchanged....
```

Subsequent S*x* call:

```
Command = Sx
Command ID = SX01
ISN Lower Limit = 0
```

```
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7

Resulting ISN buffer:

```
8    12  14  15  24
```

Example 3 : With ISN Overflow Handling

Initial S*x* call (save-ISN-list option not used):

```
Command = Sx
Command ID = SX02
Command Option 1 = blank     (no option)
ISN Lower Limit = 0
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total ISNs returned in ISN buffer and stored on Work)

Resulting ISN list: (only ISNs 31 and 33 are stored on Work)

```
8    12  14  15  24  31  33
```

Resulting ISN buffer:

```
8    12  14  15  24
```

Subsequent S*x* call:

```
Command = Sx
Command ID = SX02
ISN Lower Limit     (not used)
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 2

Resulting ISN buffer:

```
31   33   14   15  24
```

ISNs 31 and 33 are deleted from the Adabas Work, and command ID SX02 is released. A subsequent S*x* call with command ID 'SX02' will be processed as an initial S*x* call since 'SX02' was released after the last ISNs were returned to the user.

Although the ISN lower limit is not specified in this example, a non-zero value would also return only those ISNs greater than the specified value in the ISN buffer, just as in example 2.

Example 4 : Without ISN Overflow Handling

Initial S*x* call with blank or zero command ID:

```
Command = Sx
Command ID = blanks or binary zeros
Command Option 1 = blank     (no option)
ISN Lower Limit = 0          (no lower limit specified)
ISN Buffer Length = 20
CALL ADABAS ...
```

Resulting ISN quantity = 7 (total matching ISNs)

Resulting ISN list: none are stored on Work

```
8  12  14  15  24  31  33
```

Resulting ISN buffer:

```
8  12  14  15  24
```

A subsequent S*x* call with command ID equal to blanks or binary zeros and ISN lower limit equal to 0 will result in a reexecution of the same find command with the same result as the initial call. A subsequent call with command ID equal to blanks or binary zeros and ISN lower limit = 24 causes reexecution of the S*x* command. The result will be ISN quantity of 2 with ISNs 31 and 33 in the ISN buffer.

Using the Multifetch/Prefetch Feature

Programs that process large amounts of data in sequential order require frequent storage access, causing long execution times. The Adabas multifetch and prefetch options significantly reduce the execution times of such programs by reducing the number of system commands needed to complete Adabas calls.

The multifetch and prefetch options reduce execution time in almost all normal applications; however, the specific advantage depends on the type of application program.

Note: In Adabas 8, ACBX interface calls support the multifetch but not the prefetch feature. However, the prefetch feature is still supported for **ACB interface direct calls**; so, if your application uses **ACB interface direct calls**, you can continue to use the prefetch feature in those calls only.

This section covers the following topics:

- Multifetching Versus Prefetching
- Invoking Multifetch/Prefetch
- Multifetch Operation Processing

Multifetching Versus Prefetching

The multifetch and prefetch features reduce the communication overhead between the application program and the Adabas nucleus. Multifetch operations store multiple records that result from a single call and then transfer the records to the user in the **record buffer**. Without multifetch operations, multiple Adabas calls would be necessary to obtain the same result. When an **ACB interface direct call** is performed, the record descriptor elements (RDEs) of multifetched records are stored in the **ISN buffer**; when an **ACBX interface direct call** is performed, the record descriptor elements (RDEs) of multifetched records are stored in **multifetch buffers**.

Multifetch operation is similar to prefetch; multifetch comprises prefetch functions and more. Releases of Adabas prior to Adabas 8 support *both* prefetch and multifetch operations; however, new programs should use the multifetch (M) option, which is common across all Adabas platforms. In addition, the prefetch option is no longer supported for releases of Adabas 8 or later.

Invoking Multifetch/Prefetch

If you are using *only* **ACB interface direct calls** in your application, you can invoke prefetch or multifetch processing using one of two methods. How they are invoked determines where the preread records are held for processing and therefore what buffer space must be allocated.

- The first method, specifying the PREFETCH=YES (for multifetch processing) or OLD (for prefetch processing) parameter on the ADARUN statement, is the most efficient and requires no application programming changes. These parameters provide control for batch jobs.

  When PREFETCH=YES or OLD, Adabas uses a double-buffering technique that allows processing of one group of records while the following group is being fetched.

  For more information about the prefetch/multifetch ADARUN parameters, which include PREFETCH, PREFICMD, PREFIFIL, PREFNREC, PREFSBL, PREFTBL, PREFXCMD, and PREFXFIL, read *Adabas Initialization (ADARUN Statement)*, in the *Adabas Operations Manual*.

  > **Note:** PREFETCH=OLD is available for the purpose of upwards compatibility. Some newer Adabas features are no longer supported. For example, attempts to pass APLX multiple buffers with this access will return a response code 22 (ADARSP022), subcode 51. We recommend you use PREFETCH=YES if such new features must be supported.

- The second method is to specify the M or O option (for multifetch processing) in the L1/L4, L2/L5, L3/L6, L9, BT or ET commands or the P option (for prefetch processing) in the L1/L4, L2/L5, L3/L6 or L9 commands. Use of the command-level options M, O, and P are provided in the documentation for individual commands, as well as in ***Multifetch Operation Processing*** and ***Prefetch Operation Processing***, elsewhere in this section.

If you use **ACBX interface direct calls** in your application or if you *mix* ACB and ACBX interface direct calls in your application, you can invoke multifetch processing only by specifying the M or O option in the L1/L4, L2/L5, L3/L6, L9, BT or ET commands. Use of the command-level options M and O are provided in the documentation for individual commands, as well as in ***Multifetch Operation Processing***, elsewhere in this section.

Note: Prefetch processing is not supported for applications that use **ACBX interface direct calls**.

Multifetch Operation Processing

Multifetch operations are compatible with the corresponding operations on non-mainframe platforms, and can be used across platforms in heterogeneous environments.

Multifetching can be used with the following Adabas commands:

- L1/L4 with I or N option (read by ISN, find with GET NEXT)
- L2/L5 (read physical)
- L3/L6 (read logical by descriptor)
- L9 (histogram)
- BT (backout transaction)
- ET (end of transaction)

For all read calls (L*x*), multifetch returns a group of records in the record buffer and a description of these records in either the caller's ISN buffer (for **ACB interface direct calls**) or the caller's multifetch buffer (for **ACBX interface direct calls**). The maximum number of records is limited by the following values, which are specified in one of Adabas control blocks (ACB or ACBX) or the ABD, as appropriate:

- User-defined maximum as input to the call
- Record buffer length
- ISN buffer length (**ACB interface direct calls**)
- Multifetch buffer length (**ACBX interface direct calls**)

This section covers the following topics:

- BT / ET Multifetch Processing

READ (Lx) Multifetch Processing

If you are making a direct call for a read command and you want to use multifetch processing, certain fields must be set prior to the call, as follows:

What to Set	Where to Set It		
	ACB Interface Direct Call	ACBX Interface Direct Call	
	ACB	ACBX	ABD
Supported command type and options (see the command list in section *Multifetch Operation Processing*)	Command code (ACBCMD)	Command code (ACBXCMD)	---
Maximum number of values to return, or *0* to multifetch all values.	ISN lower limit (ACBISL)	ISN lower limit (ACBXISL)	---
Set to "M" or *0* (see notes below)	Command Option 1 (ACBCOP1)	Command Option 1 (ACBXCOP1)	---
Length of the record buffer	Record buffer length (ACBRBL)	---	Buffer size (ABDXSIZE) in the record buffer ABD
Length of the ISN buffer	ISN buffer length (ACBIBL)	---	---
Length of the multifetch buffer	---	---	Buffer size (ABDXSIZE) in the multifetch buffer ABD

Notes:

1. *Command option "M" indicates that the multifetch option is to be used. Command option "O" selects both the multifetch option ("M") and the existing command option "R" (returns Adabas response code 145 (ADARSP145) for a requested ISN that is already being held by another user) for the L4/L5/L6 commands.*
2. *For an L1 command, either the command option "I" (ISN sequence) or option "N" (GET NEXT option) must be specified with the multifetch command option "M" or "O"; otherwise, Adabas response code 22 (ADARSP022) occurs.*

The contents of the returned record buffer and ISN or multifetch buffer are as follows:

```
Record Buffer: record1,record2, ... ,recordn
```

Records are returned in the record buffer as usual. If more than one record is returned, all records are placed adjoining in the record buffer.

Descriptive elements for these records are returned in the ISN or multifetch buffer. The first (leftmost) fullword of the ISN or multifetch buffer contains the number of elements that follow (signed integer, four bytes). Following this count are the record descriptor elements, each 16 bytes long:

```
ISN or Multifetch Buffer: RDE count{RDE1 }...
```

A record descriptor element (RDE) has the structure shown in the following table.

Format	Length	Content
All fields unsigned integer, right aligned	4 bytes	Length of this record in record buffer. Records may have different lengths.
	4 bytes	Adabas response for this record. If a nonzero response is given, no record is stored in the record buffer.
	4 bytes	ISN for this record.
	4 bytes	(L9 only) ISN quantity: value count for this descriptor.

If an error is detected while the first record is being processed, the error response is returned in the response code field of the appropriate Adabas control block (ACBRSP or ACBXRSP).

If an error is detected while a record other than the first is being processed, the response code is returned in the corresponding record descriptor element in the ISN or multifetch buffer.

BT / ET Multifetch Processing

By default, Adabas releases all currently held ISNs for the user issuing a BT/ET command. With the multifetch option, only a subset of the records held by the current transaction is released. The records to be released from hold status are specified in the ISN buffer if you are using the **ACB direct call interface**; if you are using the **ACBX direct call interface**, the records to be released from hold status are specified in the multifetch buffer. The first fullword in the ISN or multifetch buffer specifies the number of 8-byte elements following.

You can activate the command-level multifetch feature for the ET/BT command call by setting the following fields of the Adabas control block as indicated:

If you are making a direct call for an ET or BT command and you want to use multifetch processing, certain fields must be set prior to the call, as follows:

What to Set	Where to Set It		
	ACB Interface Direct Call	ACBX Interface Direct Call	
	ACB	ACBX	ABD
"BT" or "ET"	Command code (ACBCMD)	Command code (ACBXCMD)	---
"M"	Command Option 1 (ACBCOP1)	Command Option 1 (ACBXCOP1)	---
Length of the ISN buffer	ISN buffer length (ACBIBL)	---	---

What to Set	Where to Set It		
	ACB Interface Direct Call	ACBX Interface Direct Call	
	ACB	ACBX	ABD
Length of the multifetch buffer	---	---	Buffer size (ABDXSIZE) in the multifetch buffer ABD

Note: If multifetch is set with ADARUN PREFETCH=YES, the "P" option (prefetch) is automatically used for ET and BT commands; the "M" option (multifetch) is automatically used for all other commands.

The ISN or multifetch buffer must contain the following values:

```
ISN  or Multifetch Buffer: ISN descriptor count {ISN descriptor element (See table ↩
below)} ...
```

An ISN descriptor element has the structure shown in the following table.

Format	Length	Content
Binary, right aligned	4 bytes	Adabas file number
	4 bytes	ISN

Prefetch Operation Processing

Prefetch is effective for programs that call sequential commands (L1/L4 with GET NEXT, L2/L5, L3/L6, L9) using the **ACB direct call interface**. It is not available for use with **ACBX direct calls**.

When using prefetch, a series of sequential read commands requires only one Adabas call. This single call causes several records to be read at a time from the database. This results in a significant reduction in interregion communication overhead and also permits the overlapped operation of the user program and the Adabas nucleus.

Note: If the hold option is used (L4/5/6 commands), Adabas places records in hold status when they are read into the prefetch buffer area. This means that if an ET command is issued before all records have been processed, all records (including those not yet processed) are released. The hold ISN option of the ET or HI command can be used to place any such records back into hold status.

Specific commands or files can be excluded from prefetch option processing by specifying the files or commands to be excluded with the respective ADARUN PREFXFIL or PREFXCMD parameters.

This section covers the following topics:

- Invoking Prefetch Operation with Command Option P

- Additional Prefetch Programming Considerations

Invoking Prefetch Operation with Command Option P

When enabling prefetch with the command-specific P option, Adabas uses the ISN buffer defined within the user program as the intermediate storage area for the pre-read records. Each record in the ISN buffer is preceded by a 16-byte header:

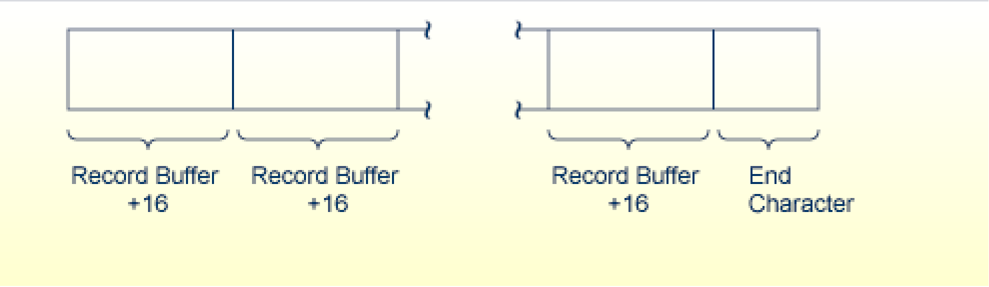
Byte	Use
1-2	Length of record (including length definition). A length of zero indicates the end of data.
3-4	Nucleus response code
5-8	Nucleus internal ID (if the response code is neither zero nor 3, a subcode is returned in the rightmost 2 bytes)
9-12	ISN of the record
13-16	ISN quantity (L9 command only)

The first record is provided by Adabas in the record buffer (without the 16-byte header). The user must then process additional records from the ISN buffer. When end-of-file occurs, the header of the last record in the ISN buffer contains Adabas response code 3 (ADARSP003), and the two-byte end character contains binary zeros.

Additional Prefetch Programming Considerations

The following are points to consider when using the prefetch option:

- The record buffer size should be set just large enough to contain the largest expected decompressed record.
- If the sequential pass of a file is not to be continued until end-of-file condition is detected, be sure to issue an RC command to release the command ID used whenever file processing has been completed.
- The command ID should not be changed during file processing.
- When using a command option "P" to invoke prefetch operation, the ISN buffer size must be a multiple of the total of the record buffer length plus 16, and a final two bytes for an end character:



ISN Buffer Size for Prefetch Programming

The Effects of ISN Changes on Prefetched or Multifetched Records

If a prefetched or multifetched record is updated, the following processing is applied:

> **Important:** You should not modify a file (especially the descriptor being accessed) while reading or locating (finding) records in the same file.

- Any update-protection only applies to the active session; anything done by other sessions is not included.
- If an ISN is modified for a record that has been prefetched or multifetched, it is re-fetched (via the L1 command) when the program finally gets to it.
- If an ISN is deleted for a record that has been prefetched or multifetched, it is skipped.
- If an ISN is inserted for a record that has been prefetched or multifetched, it is skipped. For this reason, we do not recommend that you modify files while reading or locating records in the same file.

II Calling Adabas

5 Calling Adabas

This chapter describes the available procedures you can use to call Adabas to execute an Adabas command. Adabas direct calls use the standard calling procedure provided by the host language (for example, Assembler, COBOL, Fortran, C, or PL/I).

Note: Examples of Adabas calls in a variety of host languages are provided with the programming examples in ***Programming Examples***, elsewhere in this guide.

There are two kinds of Adabas direct calls, one for each of the different control block interfaces supported by Adabas:

- The *ACB direct call interface* is the classic direct call interface, used for Adabas releases prior to Adabas version 8. Direct calls in this format require the use of the classic Adabas control block (ACB). If you have been using releases of Adabas prior to Adabas 8, the direct calls used by your applications use the ACB direct call interface.
- The *ACBX direct call interface* is the extended direct call interface, used for Adabas releases starting with Adabas 8. Direct calls in this format require the use of the *extended* Adabas control block (ACBX). If you have purchased and installed Adabas 8 (or later), you can use this format of direct call in your applications. Otherwise, you cannot.

Adabas version 8 fully supports *both* the ACB and the ACBX direct call interfaces:

- Existing application programs that use the ACB direct call interface can continue to run in the same way, without change.
- In addition, you can decide whether you want to use the ACBX-based or ACB-based direct call interface in your application programs, on a call-by-call basis. The same program can use both interfaces.

The control block and the related buffers specify which Adabas command is to be executed and provide any additional information (parameters or operands) required for the command. The pointer to the appropriate control block (ACB or ACBX) must always be the first operand specified in an Adabas call.

How Adabas Distinguishes Between ACB and ACBX Direct Calls

Any application program can make both ACB and ACBX direct calls. The control block (ACB or ACBX) is the first parameter in Adabas calls using either the ACB or ACBX interfaces. Adabas 8 determines which control block is used for a call by the presence of a value starting with the letter "F" at offset 2 of the control block. Offset 2 in the ACB is the command code field (ACBCMD), but since there is no valid F* Adabas command, no valid direct call using the ACB will contain a value starting with the letter "F" at offset 2. Offset 2 in the ACBX is a new version field (ACBXVER) identifying the new ACBX.

The presence or absence of an "F" at offset 2 determines how Adabas 8 interprets the direct call. If an "F" is specified in offset 2, Adabas interprets the control block and remaining direct call parameters as an ACBX call; if an "F" is *not* specified in offset 2, Adabas interprets the control block and remaining direct call parameters as an ACB call. If, for some reason, the remaining control block fields and direct call parameters are *not* specified correctly for the type of call indicated by the presence or absence of an "F" at offset 2 (for example, if ACB parameters are specified for an ACBX call), errors may result or the results of the call may not be as expected. For more information about how direct calls are specified using the ACB or the ACBX, read ***Specifying an ACB Interface Direct Call*** or ***Specifying an ACBX Interface Direct Call***, elsewhere in this guide.

Specifying an ACB Interface Direct Call

When making a direct call using the ACB interface, syntax such as the following should be used (this is a COBOL example):

```
CALL 'ADABAS' USING acb-control-block-name
                    [format-buffer]
                    [record-buffer]
                    [search-buffer]
                    [value-buffer]
                    [ISN-buffer]
```

In an ACB direct call, Adabas expects buffers to be specified in the order shown in this syntax. If no buffers are required for a call, no buffers need be specified. However, if a given call does not require a format buffer, but does require one of the other buffers (for example, a record buffer), a dummy (or blank) format buffer must be specified prior to the record buffer. Likewise, if a call requires only an ISN buffer, dummy format, record, search, and value buffers must be supplied as well.

The following table describes each of the italicized, replaceable items in this syntax. For more information about the format of the ACB control block and Adabas buffers, read ***Adabas Control Block (ACB)*** and ***Defining Buffers***, elsewhere in this guide. For information about the relationships between different ABD types, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

Replace	With
acb-control-block-name	The pointer to the Adabas Control Block (ACB) to use for the call.
format-buffer	The name of or pointer to the format buffer to use for the call. Only one format buffer can be specified in a single ACB direct call.
ISN-buffer	The name of or pointer to the ISN buffer to use for the call. Only one ISN buffer can be specified in a single ACB direct call.
record-buffer	The name of or pointer to the record buffer to use for the call. Only one record buffer can be specified in a single ACB direct call.
search-buffer	The name of or pointer to the search buffer to use for the call. Only one search buffer can be specified in a single ACB direct call.

Replace	With
value-buffer	The name of or pointer to the value buffer to use for the call. Only one value buffer can be specified in a single ACB direct call.

Specifying an ACBX Interface Direct Call

The way direct calls are made in your applications when using the new ACBX interface is different than when using the classic ACB interface. In addition, the calls are different for mainframe applications and open systems applications. This section covers the following topics:

- Specifying an ACBX Interface Direct Call in Mainframe Applications
- Specifying an ACBX Interface Direct Call in Open System Applications

Specifying an ACBX Interface Direct Call in Mainframe Applications

The way direct calls are made in your applications when using the new ACBX interface is different than when using the classic ACB interface. When making a direct call using the ACBX interface in mainframe applications, syntax such as the following should be used (this is a COBOL example):

```
CALL 'ADABAS' USING acbx-control-block-name
                    reserved-fullword
                    reentrancy-token
                  [format-buffer-ABD record-buffer-ABD [multifetch-buffer-ABD]]...
                    [search-buffer-ABD]
                    [value-buffer-ABD]
                    [ISN-buffer-ABD]
                    [performance-buffer-ABD]
                    [user-buffer-ABD]
```

Each ABD either directly precedes its associated buffer or contains a pointer to the buffer. It effectively represents the buffer.

ABDs can be specified in any sequence in an ACBX interface direct call. However, if an ABD requires a matching ABD of another type, Adabas will match them sequentially. For example, if three format buffer ABDs and three record buffer ABDs are included in the call, the first format buffer ABD in the call is matched with the first record buffer ABD in the call, the second format buffer ABD is matched with the second record buffer ABD, and the third format buffer ABD is matched with third record buffer ABD.

If unequal numbers of match-requiring ABDs are specified, Adabas will generate a dummy ABD (with a buffer length of zero) for the missing ABD. For example, if three format buffer ABDs are specified, but only two record buffer ABDs are specified, a dummy record buffer ABD is created for use with the third format buffer ABD. If you would prefer that the dummy record buffer ABD be used for the second format buffer ABD instead, you must specify the dummy record buffer ABD yourself prior to the record buffer ABD to be used by the third format buffer ABD.

For commands where data in the record buffer is *not* described by a format specification in the format buffer, no format buffer segments need be specified; if any are specified, they are ignored. This applies to only a few commands; the most prominent of them is OP.

The following table describes each of the italicized, replaceable items in this syntax. For more information about the format of the extended Adabas control block (ACBX), Adabas buffer descriptions (ABDs), and Adabas buffers, read ***Extended Adabas Control Block (ACBX)***, ***Adabas Buffer Descriptions (ABDs)***, and ***Defining Buffers***, elsewhere in this guide. For information about the relationships between different buffer types, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

Replace	With
acbx-control-block-name	The pointer to the extended Adabas control block (ACBX) to use for the call.
format-buffer-ABD	The name of or pointer to the format buffer ABD that defines a format buffer segment to use for the call. Each format buffer segment must end with a period and be a complete and valid standalone format buffer. Multiple format buffer ABDs can be specified in a single ACBX direct call.
ISN-buffer-ABD	The name of or pointer to the ISN buffer ABD that defines an v segment to use for the call. Only one ISN buffer ABD can be specified in a single ACBX direct call.
multifetch-buffer-ABD	The name of or pointer to the multifetch buffer ABD that defines a multifetch buffer segment to use for the call. Multiple multifetch buffer ABDs can be specified in a single ACBX direct call.
performance-buffer-ABD	The name of or pointer to the performance buffer ABD that defines a performance buffer segment used by Adabas Review. The performance buffer segment is reserved for use by Adabas Review.
record-buffer-ABD	The name of or pointer to the record buffer ABD that defines a record buffer segment to use for the call. Multiple record buffer ABDs can be specified in a single ACBX direct call.
reentrancy-token	The ADALNK reentrancy token. This is a fullword in the calling program's storage where ADALNK stores the address of its static data area. This fullword should be set to zero before the first Adabas call. It should then remain unchanged for all subsequent direct calls while the program runs.
reserved-fullword	The fullword containing binary zeros. This fullword is reserved for use by Adabas and should be set to binary zeros before the first Adabas call.
search-buffer-ABD	The name of or pointer to the search buffer ABD that defines a search buffer segment to use for the call. Only one search buffer ABD can be specified in a single ACBX direct call.
user-buffer-ABD	The name of or pointer to the user buffer ABD that defines a user buffer segment (extension) to use for the call. The user buffer extension (UBX) is used for the user data passed to user exits LNKUEX1 (link routine pre-call exit) and LNKUEX2 (link routine post-call exit). A single user buffer ABD can be specified in an ACBX direct call.

Replace	With
value-buffer-ABD	The name of or pointer to the value buffer ABD that defines a value buffer segment to use for the call. Only one value buffer ABD can be specified in a single ACBX direct call.

Specifying an ACBX Interface Direct Call in Open System Applications

The way direct calls are made in your applications when using the new ACBX interface is different than when using the classic ACB interface. When making a direct call using the ACBX interface in open system applications, syntax such as the following should be used (this is a COBOL example):

```
CALL 'ADABAS' USING acbx-control-block-name
                    ABD-count
                    ABD-list-pointer
```

The following table describes each of the italicized, replaceable items in this syntax. For more information about the format of the extended Adabas control block (ACBX), Adabas buffer descriptions (ABDs), ABD lists, and Adabas buffers, read ***Extended Adabas Control Block (ACBX)***, ***Adabas Buffer Descriptions (ABDs)***, ***ABD Lists***, and ***Defining Buffers***, elsewhere in this guide.

Replace	With	Conditions
acbx-control-block-name	The pointer to the extended Adabas control block (ACBX) to use for the call.	Required.
ABD-count	The number of ABD pointers included in the ABD list for the direct call.	Required only if ABDs and their associated buffers are used in the direct call.
ABD-list-pointer	The pointer to the ABD list for the direct call. The ABD list contains pointer references for all of the ABDs used by the ACBX direct call. For more information about the ABD list, read *ABD Lists*, elsewhere in this guide.	Required only if buffers are required for the direct call.

Mixing ACB and ACBX Direct Calls

You can freely mix ACB and ACBX direct calls in the same application.

In TSO or batch environments, when Adabas 8 non-reentrant (ADALNK) direct calls are invoked using both the ACB and ACBX direct call interfaces in the same application, user context is preserved because the work area used for the calls is part of the ADALNK module itself. However, if you elect to use reentrant (ADALNKR) direct calls using both the ACB and ACBX direct call interfaces in the same application, you must ensure that the user context is preserved yourself, or your application may produce incorrect results. For information on preserving user context in ADALNKR direct calls, read *Mixing ACB and ACBX Interface Direct Calls to ADALNKR*, in *Adabas Operations Manual*.

III Adabas Control Block Structures (ACB and ACBX)

6 Adabas Control Block Structures (ACB and ACBX)

Two kinds of control blocks are now supported by Adabas:

- The *Adabas control block (ACB)* is the classic control block, used for Adabas releases prior to Adabas version 8. If you have been using releases of Adabas prior to Adabas 8, the direct calls used by your applications use the ACB. It is important to note that Adabas 8 fully supports the ACB, so you are not required to update your existing applications once you install Adabas 8.
- The *extended Adabas control block (ACBX)* can be used in Adabas releases starting with Adabas 8. The ACBX supports the increased buffer sizes and segmented buffers introduced in Adabas 8. If you have purchased and installed Adabas 8 (or later), you can use the ACBX in direct calls from your applications. Otherwise, you cannot.

The use of each applicable field in the control blocks is explained with each Adabas command in ***Commands***, elsewhere in this guide. To ensure user program compatibility with later Adabas releases, all control block fields not used by a particular command should be set to zeros or blanks, depending on field type.

The position of each field in a control block is fixed. In addition, all values in the control block must be entered in the data type defined for the field. For example, the ISN field is defined as binary format; therefore, any entry made in this field must be in binary format.

Notes:

1. Adabas and other Software AG program products use some control block fields for internal purposes, and may return values in some fields that have no meaning to the user. These uses and values may be release-dependent, and are not appropriate for program use. Software AG therefore recommends that you use only the fields and values described in this documentation. *In addition, you should always initialize unused control block fields with either zeros or blanks, according to their field types.*
2. Some Adabas-dependent Software AG products return control block values such as response codes and subcodes. Refer to the documentation for those products for a description of the product-specific control block values.

Adabas Control Block (ACB)

The Adabas control block (ACB) is 80 bytes long . This section covers the following topics:

- ACB Examples

ACB Format

The following table describes the format of the ACB. We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

DSECT Name	Field	Control Block Position	Offset	Length (in Bytes)	Format
ACBTYPE	Call Type	1	00	1	binary
reserved	(reserved)	2	01	1	binary
ACBCMD	Command Code	3-4	02	2	alphanumeric
ACBCID	Command ID	5-8	04	4	alphanumeric / binary
ACBFNR	File Number	9-10	08	2	binary
ACBRSP	Response Code	11-12	0A	2	binary
ACBISN	ISN	13-16	0C	4	binary
ACBISL	ISN Lower Limit	17-20	10	4	binary
ACBISQ	ISN Quantity	21-24	14	4	binary
ACBFBL	Format Buffer Length	25-26	18	2	binary
ACBRBL	Record Buffer Length	27-28	1A	2	binary
ACBSBL	Search Buffer Length	29-30	1C	2	binary
ACBVBL	Value Buffer Length	31-32	1E	2	binary
ACBIBL	ISN Buffer Length	33-34	20	2	binary
ACBCOP1	Command Option 1	35	22	1	alphanumeric
ACBCOP2	Command Option 2	36	23	1	alphanumeric
ACBADD1	Additions 1	37-44	24	8	alphanumeric / binary
ACBADD2	Additions 2	45-48	2C	4	alphanumeric / binary
ACBADD3	Additions 3	49-56	30	8	alphanumeric
ACBADD4	Additions 4	57-64	38	8	alphanumeric
ACBADD5	Additions 5	65-72	40	8	alphanumeric / binary
ACBCMDT	Command Time	73-76	48	4	binary
ACBUSER	User Area	77-80	4C	4	not applicable

ACB Fields

The content of the control block fields and buffers must be set before an Adabas command (call) is issued. Adabas also returns one or more values or codes in certain fields and buffers after each command is executed.

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Each of the fields in the ACB is described in this section, in the order they appear in the **ACB format**. The descriptions are valid for most Adabas commands; however, some Adabas commands use some control block fields for purposes other than those described here. For complete information about how these fields are used by each Adabas command, read ***Commands***, elsewhere in this guide.

Call Type (ACBTYPE)

The first byte of the Adabas control block (ADACB) is used by the Adabas API to determine the processing to be performed. For more information, read *Linking Applications to Adabas* in *Adabas Operations Manual*.

The values for logical requests are:

Hex	Indicates ...
X'00'	a 1-byte file number (file numbers between 1 and 255) or DBID.
X'30'	a 2-byte file number (file numbers between 1 and 65535) or DBID.
X'40'	values greater than or equal to a blank. These are accepted as "logical application calls" to maintain compatibility with earlier releases of Adabas.

Note: The X'44', X'48', and X'4C' calls are reserved for use by Software AG and are therefore *not* accepted.

All other values in the first byte of the ADACB are reserved for use by Software AG.

Because an application can reset the value in the first byte of the ADACB on each call, it is possible to mix both one- and two-byte file number (DBID) requests in a single application. In this case, you must ensure the proper construction of the file number (ACBFNR) and response code (ACBRSP) fields in the ADACB for each call type. See the discussions of these fields for more information.

Software AG recommends that an application written to use two-byte file numbers always place X'30' in the first byte of the ADACB, the logical database ID in the ACBRSP field, and the file number in the ACBFNR field. The application can then treat both the database ID and file number as 2-byte binary integers, regardless of the value for the file number in use.

Applications written in Software AG's Natural language need not include this first byte of the Adabas ACB because Natural supplies appropriate values.

Command Code (ACBCMD)

The command code defines the command to be executed, and comprises two alphanumeric characters (for example, OP, A1, BT).

Command ID (ACBCID)

The command ID field is used by many Adabas commands to identify logical read sequences, search results, and (optionally) decoded formats for use by subsequent commands. You can specify alphanumeric or binary command IDs as you choose or you can request the generation of new binary command IDs by Adabas. See the section *General Programming Considerations* for more information about command IDs. For ET, CL, and some OP commands, Adabas returns a binary transaction sequence number in the command ID field.

File Number (ACBFNR)

Note: For commands that operate on a coupled file pair, this field specifies the primary file from which ISNs or data are returned.

The file number may be one or two bytes.

Single-byte File Numbers and DBIDs

For an application program issuing Adabas commands for file numbers between 1 and 255 (single byte), build the control block as follows:

Position	Action
1	Place X'00' in the first byte of the ADACB.
9	Place the file number in the second (rightmost) byte of the ACBFNR field of the ADACB. The first (leftmost) byte of the ACBFNR field is used to store the logical (database) ID or number.

If the first byte in ACBFNR is set to zero (B'0000 0000'), the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data, or the default database ID value assembled into the link routine at offset X'80'.

Double-byte File Numbers and DBIDs

Adabas permits the use of file numbers greater than 255 on logical requests. For an application program issuing Adabas commands for file numbers between 256 and 5000 (two-byte), build the control block as follows:

Position	Action
1	Place X'30' in the first byte of the ADACB.
9	Use both bytes in ACBFNR for the file number, and use the two bytes in ACBRSP for the database (logical) ID.

If the ACBRSP field is zero, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data, or the default database ID value assembled into the link routine at offset X'80'.

Response Code (ACBRSP)

The response code field is used for two-byte database IDs.

It is also always set to a value when the Adabas command is completed. Successful completion is normally indicated by a response code of zero. For repeatable commands that process sequences of records or ISNs, other response codes indicate end-of-file or end-of-ISN-list. Non-zero response codes are defined in the *Adabas Messages and Codes*.

ISN (ACBISN)

The ISN field both specifies a required four-byte Adabas ISN value required by the command and, where appropriate, returns either the first ISN of a command-generated ISN list, or an ISN of the record read by the command.

ISN Lower Limit (ACBISL)

ISN lower limit specifies the starting point in an ISN list or range where processing is to begin. When using the multifetch option, this field holds an optional maximum count of prefetched records to return; if zero, there is no limit.

Following successful OP command completion, Adabas returns its system, call type, and nucleus ID (nucid) information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 4 and 5. See the section ***Values Returned in Control Block Fields*** for detailed information.

ISN Quantity (ACBISQ)

The ISN quantity field is a count of ISNs returned by a command. The count can be a total of all ISNs in an ISN list, or the total ISNs entered into the ISN buffer from a larger pool of ISNs by this operation.

Following successful OP command completion, Adabas returns its system release information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 6 and 7. See the section ***Values Returned in Control Block Fields*** for detailed information.

In addition, S*x* commands using security-by-value set this field to 1 when *more than one* ISN meet the search criteria.

Buffer Length: Format, Record, Search, Value, and ISN (ACBFBL, ACBRBL, ACBSBL, ACBVBL, and ACBIBL)

The format, record, search, value, and ISN buffer length fields specify the size of the related buffers. A buffer's size usually remains the same throughout a transaction. In some ISN-related operations, the ISN buffer size value determines how a command processes ISNs; for example, specifying a zero ISN buffer length causes some commands to store a resulting ISN list in the Adabas work area. If a buffer is not needed for an Adabas command, the corresponding length value should be set to zero. In some cases (multifetch option, as an example), there is a limit on the length of the buffer; see the specific command descriptions for more information.

Command Option 1 and Command Option 2 (ACBCOP1 and ACBCOP2)

The Command Option 1 and 2 fields allow you to specify processing options (ISN hold, command-level prefetching control, returning of ISNs, and so on).

Additions 1 (ACBADD1)

The Additions 1 field sometimes requires miscellaneous command-related parameters such as qualifying descriptors for creating ISN lists, or the second file number of a coupled file pair.

Additions 2 (ACBADD2)

The Additions 2 field returns the compressed record length in the leftmost (high-order) two bytes and decompressed length of record buffer-selected fields in the rightmost (low-order) two bytes for all A*n*, L*n*, N*n*, and S1/2/4 commands. The OP (open) and RE (read ET data) commands return transaction sequence numbers in this field. If Entire Net-work is installed, some response codes return the node ID of the "problem" node in the leftmost two bytes of the Additions 2 field.

If a command results in a nucleus response code, the Additions 2 field's low-order (rightmost) two bytes (47 and 48) can contain a hexadecimal subcode to identify the cause of the response code. For example, if no OP command began the session and the ADARUN statement specified OPENRQ=YES, a response code 9 (ADARSP009), subcode 66 is returned and these bytes are set to the hexadecimal value 0042 corresponding to the decimal 66. Response codes and their subcodes (as decimal equivalents) are described in the section *Nucleus Response Codes* in the *Adabas Messages and Codes*.

Note: If you are running with Entire Net-Work, the leftmost two bytes of the ACB Additions 2 field (ACBADD2) may contain the ID of the Entire Net-Work node that issued this response code.

Additions 3 (ACBADD3)

The Additions 3 field is for providing a user`s password for accessing password-protected files. If the file containing the field is actually password-protected, the password in this field is replaced with spaces (blanks) during command execution before Adabas returns control to the user program.

Additions 4 (ACBADD4)

The Additions 4 field must be set to a cipher code for those instructions that read or write encrypted (ciphered) database data files. For commands requiring multiple command IDs but no cipher code, one of the command IDs is specified in this field.

When processed by the nucleus, an Adabas call returns the Adabas release (version and revision) level numbers and the database ID in the low-order (rightmost) *three* bytes of the Additions 4 field with the format `vrnnnn` where

`v` is the Adabas version number;

`r` is the Adabas revision level number; and

`nnnn` is the number (hexadecimal) of the Adabas database that processed the call.

For example, "741111" indicates that an Adabas version 7.4 nucleus on database 4369 processed the call.

Additions 5 (ACBADD5)

The high-order (leftmost) two bits of the first byte of the Additions 5 field control the unique or global format ID selection; the low-order (rightmost) four or eight bytes can contain either an optional unique or global format ID, respectively. Refer to the section ***Using a Global Format ID*** for a complete description of this feature. A global format ID to be deleted can be specified in this field for the RC (release command ID) command. When completed, the OP command returns any optionally specified non-activity and/or transaction timeout values in the Additions 5 field.

Command Time (ACBCMDT)

The command time field is used by Adabas to return the elapsed time which was needed by the nucleus to process the command. This does not include the times when the thread was waiting on Adabas I/O operations or other resources. The time, counted in 16-microsecond units, is called "Adabas thread time". The returned count is in binary format.

User Area (ACBUSER)

The user area field is reserved for use by the user program. When making logical user calls, the user area is neither written nor read by Adabas.

For compatibility with future Adabas releases, Software AG recommends that you set unused control block fields to null values corresponding to the field's data type.

ACB DSECT

The ACB DSECT can be found in member ADACB of the distributed Adabas SRCE library.

ACB Examples

Programming examples that show control block construction in a variety of host languages are provided in section *Programming Examples* of this documentation.

- *Assembler Examples*
- *COBOL Examples*
- *PL/ I Examples*
- *FORTRAN Examples*

Extended Adabas Control Block (ACBX)

The extended Adabas control block , the ACBX, supports the increase in the buffer sizes in Adabas commands. It is 192 bytes in length (versus the 80 bytes used by the ACB). The existing, non-extended Adabas Control Block (ACB) is still supported and your existing applications will still work, but if you want to take advantage of some of the extended features provided in Adabas 8, you must use the new ACBX. Specifically, you must use the ACBX if you are using the long buffer (buffers longer than 32K) or segmented buffer (multiple format/record buffer pairs or format/record/multifetch buffer triplets) features of Adabas 8.

Otherwise, your application programs may freely switch between Adabas calls using the existing direct call interface (ACB) and calls using the new interface (ACBX).

- ACBX DSECT

ACBX Format

The following table describes the format of the ACBX. We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
ACBXTYP	Call Type	1	00	1	binary
ACBXRSV1	Reserved 1	2	01	1	binary
ACBXVER	Version Indicator	3-4	02	2	binary
ACBXLEN	ACBX Length	5-6	04	2	binary
ACBXCMD	Command Code	7-8	06	2	alphanumeric
ACBXRSV2	Reserved 2	9-10	08	2	binary
ACBXRSP	Response Code	11-12	0A	2	binary
ACBXCID	Command ID	13-16	0C	4	alphanumeric/ binary
ACBXDBID	Database ID	17-20	10	4	numeric
ACBXFNR	File Number	21-24	14	4	numeric
ACBXISNG	8-Byte ISN	25-32	18	8	do not use
ACBXISN	ISN	29-32	1C	4	binary
ACBXISLG	8-Byte ISN Lower Limit	33-40	20	8	do not use
ACBXISL	ISN Lower Limit	37-40	24	4	binary
ACBXISQG	8-Byte ISN Quantity	41-48	28	8	do not use
ACBXISQ	ISN Quantity	45-48	2C	4	binary
ACBXCOP1	Command Option 1	49	30	1	alphanumeric
ACBXCOP2	Command Option 2	50	31	1	alphanumeric
ACBXCOP3	Command Option 3	51	32	1	alphanumeric
ACBXCOP4	Command Option 4	52	33	1	alphanumeric
ACBXCOP5	Command Option 5	53	34	1	alphanumeric
ACBXCOP6	Command Option 6	54	35	1	alphanumeric
ACBXCOP7	Command Option 7	55	36	1	alphanumeric
ACBXCOP8	Command Option 8	56	37	1	alphanumeric
ACBXADD1	Additions 1	57-64	38	8	alphanumeric/ binary
ACBXADD2	Additions 2	65-68	40	4	binary
ACBXADD3	Additions 3	69-76	44	8	alphanumeric/ binary
ACBXADD4	Additions 4	77-84	4C	8	alphanumeric
ACBXADD5	Additions 5	85-92	54	8	alphanumeric/ binary

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
ACBXADD6	Additions 6	93-100	5C	8	alphanumeric/ binary
ACBXRSV3	Reserved 3	101-104	64	4	binary
ACBXERRG	Error Offset in Buffer (64-bit)	105-112	68	8	do not use
ACBXERRA	Error Offset in Buffer (32-bit)	109-112	6C	4	binary
ACBXERRB	Error Character Field	113-114	70	2	alphanumeric
ACBXERRC	Error Subcode	115-116	72	2	binary
ACBXERRD	Error Buffer ID	117	74	1	alphanumeric
ACBXERRE	Reserved for future use	118	75	1	do not use
ACBXERRF	Error Buffer Sequence Number	119-120	76	2	numeric
ACBXSUBR	Subcomponent Response Code	121-122	78	2	binary
ACBXSUBS	Subcomponent Response Subcode	123-124	7A	2	binary
ACBXSUBT	Subcomponent Error Text	125-128	7C	4	alphanumeric
ACBXLCMP	Compressed Record Length	129-136	80	8	binary
ACBXLDEC	Decompressed Record Length	137-144	88	8	binary
ACBXCMDT	Command Time	145-152	90	8	binary
ACBXUSER	User Area	153-168	98	16	not applicable
ACBXRSV4	Reserved 4	169-193	A8	24	*do not touch*

ACBX Fields

The content of the control block fields and buffers must be set before an Adabas command (call) is issued. Adabas also returns one or more values or codes in certain fields and buffers after each command is executed.

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Each of the fields in the ACBX is described in this section, in the order they appear in the **ACBX format**. The descriptions are valid for most Adabas commands; however, some Adabas commands use some control block fields for purposes other than those described here. For complete information about how these fields are used by each Adabas command, read ***Commands***, elsewhere in this guide.

Call Type (ACBXTYP)

The first byte of the Adabas control block (ADACBX) is used by the Adabas API to determine the processing to be performed. See *Linking Applications to Adabas* in the *Adabas Operations* documentation for more information.

When issuing an Adabas command, set this field to binary zeros. This indicates that a logical user call is being made (ACBXTUSR equate).

The following values in ACBXTYPE are reserved for use by Software AG and are therefore not accepted by application programs: X'04', X'08', X'0c', X'10', X'14', X'18', X'1c', X'20', X'24', X'28', X'2c', X'34', X'38', X'3c', X'44', X'48', and X'4c'.

Applications written in Software AG's Natural language need not include this first byte of the Adabas ACBX because Natural supplies appropriate values.

Reserved 1 (ACBXRSV1)

This field is reserved. Set this field to zero.

Version Indicator (ACBXVER)

The version indicator identifies whether the Adabas control block uses the new ACBX or the classic ACB format. If this field is set to a value starting with the letter "F" (for example "F2"), Adabas treats the Adabas control block as though it is specified in the ACBX format. If this field is set to any other value, Adabas treats the control block as though it is specified in the classic ACB format.

ACBX Length (ACBXLEN)

The ACBX length field should be set to the length of the ACBX structure passed to Adabas (the ACBXQLL equate, currently 192).

Command Code (ACBXCMD)

The command code defines the command to be executed, and comprises two alphanumeric characters (for example, OP, A1, BT).

Reserved 2 (ACBXRSV2)

This field is reserved. Set this field to zero.

Response Code (ACBXRSP)

This field gets set to a value when the Adabas command is completed. Successful completion is normally indicated by a response code of zero. For repeatable commands that process sequences of records or ISNs, other response codes indicate end-of-file or end-of-ISN-list. Non-zero response codes are defined in the Adabas Messages and Codes documentation found in the *Adabas Messages and Codes Guide*.

Command ID (ACBXCID)

The command ID field is used by many Adabas commands to identify logical read sequences, search results, and (optionally) decoded formats for use by subsequent commands. You can specify alphanumeric or binary command IDs as you choose or you can request the generation of new binary command IDs by Adabas. See the section ***General Programming Considerations***, elsewhere in this guide, for more information about command IDs. For ET, CL, and some OP commands, Adabas returns a binary transaction sequence number in the command ID field.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

Note: For commands that operate on a coupled file pair, this field specifies the primary file from which ISNs or data are returned.

ISN (ACBXISNG/ACBXISN)

The ISN field specifies any required Adabas ISN value required by the command and, where appropriate, returns either the ISN of the record read by the command , or the first ISN of an ISN list generated by the command.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit (ACBXISLG/ACBXISL)

ISN lower limit specifies the starting point in an ISN list or range where processing is to begin. When using the multifetch option, this field holds an optional maximum count of prefetched records to return; if zero, there is no limit.

Following successful OP command completion, Adabas returns its system, call type, and nucleus ID (nucid) information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 4 and 5. See the section ***Values Returned in Control Block Fields*** for detailed information.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

ISN Quantity (ACBXISQG/ACBXISQ)

The ISN quantity field is a count of ISNs returned by a search (S*x*) command. The count can be a total of all ISNs in an ISN list, or the total ISNs entered into the ISN buffer from a larger pool of ISNs by this operation.

Following successful OP command completion, Adabas returns its system release information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 6 and 7. See the section ***Values Returned in Control Block Fields*** for detailed information.

In addition, S*x* commands using security-by-value set this field to 1 when *more than one* ISN meet the search criteria.

The ACBXISQ field is a four-byte binary field embedded in the eight-byte ACBXISQG field, which is not yet used. Set the high-order part of the ACBXISQG field to binary zeros.

Command Options 1 through 8 (ACBXCOP1 through ACBXCOP8)

The Command Option 1 - 8 fields allow you to specify processing options (ISN hold, command-level prefetching control, returning of ISNs, and so on). In Adabas 8.1, only the Command Option 1 and Command Option 2 field are supported. However, the other Command Option fields are provided for potential expansion in future Adabas releases.

Additions 1 (ACBXADD1)

The Additions 1 field sometimes requires miscellaneous command-related parameters such as qualifying descriptors for creating ISN lists, or the second file number of a coupled file pair.

Additions 2 (ACBXADD2)

OP (open) and RE (read ET data) commands return transaction sequence numbers in this field.

Additions 3 (ACBXADD3)

The Additions 3 field is for providing a user's password for accessing password-protected files. This field is always reset to blanks during command execution.

Additions 4 (ACBXADD4)

If a command reads or writes records of an encrypted (ciphered) Adabas file, the Additions 4 field must be set to the cipher code for that file. For commands requiring multiple command IDs but no cipher code, one of the command IDs is specified in this field.

Adabas always resets this field to blanks during command execution.

When processed by a nucleus that is not running single-user mode (ADARUN MODE=SINGLE is *not* specified), the Adabas call returns the Adabas release (version and revision) level numbers and the database ID in the low-order (rightmost) *three* bytes of the Additions 4 field with the format `vrnnnn` where

`v`	is the Adabas version number;
`r`	is the Adabas revision level number; and
`nnnn`	is the number (hexadecimal) of the Adabas database that processed the call.

For example, "811111" indicates that an Adabas version 8.1 nucleus on database 4369 processed the call.

Additions 5 (ACBXADD5)

The high-order (leftmost) two bits of the first byte of the Additions 5 field control the unique or global format ID selection; the low-order (rightmost) four or eight bytes can contain either an optional unique or global format ID, respectively. A global format ID to be deleted can be specified in this field for the RC (release command ID) command. When completed, the OP command returns any optionally specified non-activity or transaction timeout values in the Additions 5 field.

Additions 6 (ACBXADD6)

This field is not used at this time. It must be set to binary zeros.

Reserved 3 (ACBXRSV3)

This field is reserved. Set this field must be set to binary zeros.

Error Offset in Buffer (64-bit) (ACBXERRG)

The Error Offset in Buffer (64-bit) and the Error Offset in Buffer (32-bit) fields specify the offset in the buffer, if any, where the error is detected during the direct call.

The Error Offset in Buffer (64-bit) field, ACBXERRG, is not yet available, but may be used in some later release. For now, use the Error Offset in Buffer (32-bit) field, ACBXERRA.

The ACBXERRx fields are only set when a response code is returned from a direct call. The ACBXERRA, ACBXERRD, and ACBXEFFE fields are only set when the response code is related to buffer processing.

Error Offset in Buffer (32-bit) (ACBXERRA)

The Error Offset in Buffer (64-bit) and the Error Offset in Buffer (32-bit) fields specify the offset in the buffer, if any, where the error is detected during the direct call.

The Error Offset in Buffer (64-bit) field, ACBXERRG, is not yet available, but may be used in some later release. For now, use the Error Offset in Buffer (32-bit) field, ACBXERRA.

The ACBXERRx fields are only set when a response code is returned from a direct call. The ACBXERRA, ACBXERRD, and ACBXERRF fields are only set when the response code is related to buffer processing.

Error Character Field (ACBXERRB)

This field identifies the two-byte Adabas short name of the field, if any, that was being processed when the error was detected.

The ACBXERRx fields are only set when a response code is returned from a direct call.

Error Subcode (ACBXERRC)

This field stores the subcode of the error that occurred during direct call processing.

The ACBXERRx fields are only set when a response code is returned from a direct call. If Entire Net-work is installed, some response codes return the node ID of the problem node in the leftmost two bytes of this field.

Error Buffer ID (ACBXERRD)

This field contains the ID (from the ABDID field) of the buffer referred to by the ACBXERRA field, so that the buffer causing the error can be identified, when multiple buffers are involved.

The ACBXERRx fields are only set when a response code is returned from a direct call. The ACBXERRA, ACBXERRD, and ACBXERRF fields are only set when the response code is related to buffer processing.

Reserved (ACBXERRE)

This field is reserved for future use. Do not use this field at this time.

Error Buffer Sequence Number (ACBXERRF)

This field contains the two-byte sequence number of the buffer segment containing the error (if any) referred to by the ACBXERRA and ACBXERRD fields.

The ACBXERRx fields are only set when a response code is returned from a direct call. The ACBXERRA, ACBXERRD, and ACBXERRF fields are only set when the response code is related to buffer processing.

Subcomponent Response Code (ACBXSUBR)

This field contains the response code from any error that occurred when an Adabas add-on product intercepts the Adabas command.

If the message was compressed using zEDC compression services and the attempt to decompress the message failed, subcode 1 is placed in the rightmost two bytes of this field.

Subcomponent Response Subcode (ACBXSUBS)

This field contains the response subcode from any error that occurred when an Adabas add-on product intercepts the Adabas command.

Subcomponent Error Text (ACBXSUBT)

This field contains the error text of any error that occurred when an Adabas add-on product intercepts the Adabas command.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written.

This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Command Time (ACBXCMDT)

The command time (also called *thread time*) field is used by Adabas to return the elapsed time that was needed by the nucleus to process the command. This does *not* include the times when the Adabas thread executing the command was waiting on Adabas I/O operations or other resources; it *does* include the times when the thread was waiting for a processor so it could execute the code. The time is measured in 1/4096 microsecond units and the returned count is in binary format.

User Area (ACBXUSER)

The user area field is reserved for use by the user program. When making logical user calls, the user area is neither written nor read by Adabas.

Reserved 4 (ACBXRSV4)

This field is reserved for use by Adabas. Your user program should set this field to binary zeros before the first Adabas call using this ACBX and then leave it unmodified thereafter.

ACBX DSECT

The ACBX DSECT can be found in member ADACBX of the distributed Adabas SRCE library. For your convenience, an **ACBX Format** table is also provided, elsewhere in this section.

Differences between the ACB and the ACBX

The ACBX differs in many ways from the ACB. The ACBX includes some fields that are not included in the ACB and the sizes of some ACBX fields are larger than their ACB equivalents. These expansions in the ACBX have been made to ensure that its structure can be flexible enough to handle potential future enhancements to Adabas, without altering its fundamental structure for many years.

This section describes the differences between the ACB and the ACBX:

- Control Block Length
- Buffer Length Fields
- Command Options, Additions, and Reserved Fields
- Unit Differences
- Field Length Differences
- Additional Fields in ACBX
- ACB Dual Purpose Field Changes

- Structure and Offset Differences

Control Block Length

The ACBX is 192 (or X'C0') bytes in length; the ACB is 80 bytes long.

Buffer Length Fields

The buffer length fields are not included in the ACBX as they are in the ACB. When using the **ACBX direct call interface**, they are instead provided in the individual Adabas buffer descriptions (ABDs). So the ACBX contains no buffer fields corresponding to the ACBFBL, ACBIBL, ACBRBL, ACBSBL, and ACBVBL found in the ACB; the ABDs associated with the call are used instead. One ABD represents an individual Adabas buffer segment. They are described in ***Adabas Buffer Descriptions***, elsewhere in this guide.

Command Options, Additions, and Reserved Fields

The number of command option, additions, and reserved control block fields are larger in the ACBX:

- The ACBX contains eight command option fields, up from the two command option fields available in the ACB.
- The ACBX contains six additions fields, up from the five additions fields available in the ACB.
- The ACBX contains four reserved fields, up from one reserved field available in the ACB.

  Reserved ACBX fields must be set to binary zeros; the reserved 4 field (ACBXRSV4) should be initialized to binary zeros and then left unchanged.

Unit Differences

The units used to measure command time (thread time) differ between the ACB and the ACBX. The ACB measures command time (ACBCMDT) in 16 microsecond units; the ACBX measures command time (ACBXCMDT) in 1/4096 microsecond units.

Field Length Differences

The lengths of many control block fields are larger in the ACBX. The following table summarizes these changes:

Field Title	Length	
	ACB	ACBX
File Number	2	4
Database ID	2	4
ISN	4	4
ISN Lower Limit	4	4
ISN Quantity	4	4
Compressed Record Length	4	8
Decompressed Record Length	4	8
Command Time	4	8
User Area	4	16
Format Buffer Length	2	4 (in the ABD)
Record Buffer Length	2	4 (in the ABD)
Search Buffer Length	2	4 (in the ABD)
Value Buffer Length	2	4 (in the ABD)

Additional Fields in ACBX

The following additional fields are available in the ACBX:

ACBX DSECT Name	Description
ACBXADD6	Additions 6
ACBXCOP3	Command options 3
ACBXCOP4	Command options 4
ACBXCOP5	Command options 5
ACBXCOP6	Command options 6
ACBXCOP7	Command options 7
ACBXCOP8	Command options 8
ACBXDBID	The database ID. In the ACB, the database ID is stored in the response code field (ACBRSP) for X’30’ calls and in the first byte of ACBFNR for other logical calls.
ACBXERRA	Error offset into the buffer (32-bit).
ACBXERRB	Error character field (field name).
ACBXERRC	Error subcode. In the ACB, the error subcode is stored in the Additions 2 field (ACBADD2).
ACBXERRD	Error buffer ID, if multiple buffers are involved.
ACBXERRE	Error buffer sequence number, if multiple buffers are involved.
ACBXERRG	Error offset into the buffer (64-bit) - this field is not yet supported.

ACBX DSECT Name	Description
ACBXLCMP	Compressed record length (or portion of record if the entire record is not read). In the ACB, the compressed record length is stored in the Additions 2 field (ACBADD2).
ACBXLDEC	Decompressed record length. In the ACB, the decompressed record length is stored in the Additions 2 field (ACBADD2).
ACBXLEN	The length of the ACBX, currently 192
ACBXRSV2	Reserved. The value of this field must be set to zero.
ACBXRSV3	Reserved. The value of this field must be set to zero.
ACBXRSV4	Reserved for use by Adabas.
ACBXSUBR	Subcomponent response code, used by Adabas add-on products.
ACBXSUBS	Subcomponent response subcode, used by Adabas add-on products.
ACBXSUBT	Subcomponent error text, used by Adabas add-on products.
ACBXVER	When set to C'F2', this field indicates to Adabas that the new extended ACB (ACBX) is used.

ACB Dual Purpose Field Changes

There are a number of cases where an ACB field that has multiple purposes has been split out into additional fields in the ACBX:

- In the ACB, the Response code field (ACBRSP) is used to store the database ID for X’30’ calls. For the other logical calls the one-byte database ID was stored in the first byte of the file number field, ACBFNR. The ACBX provides a Database ID field (ACBXDBID) for this purpose.
- In the ACB, the ACBADD2 field is used to retain error information for certain Adabas response codes. In the ACBX, error information fields (ACBXERR* series) are provided for this purpose.
- In the ACB, the ACBADD2 field is used to return, for a successful call, the compressed and decompressed record lengths of the processed data. In the ACBX, for a successful call, the Compressed Record field (ACBXLCMP) contains the length of the compressed data processed by Adabas and the Decompressed Record field (ACBXLDEC) contains the length of the decompressed data.

Structure and Offset Differences

The offset and sequence of ACBX fields is generally different from the corresponding ACB fields, as depicted in the following table.

Offset	ACB DSECT Field Name	ACBX DSECT Field Name
00	ACBTYPE (Call type)	ACBXTYPE (Call type)
01	reserved	ACBXRSV1 (reserved 1)
02	ACBCMD (Command code)	ACBXVER (ACBX version indicator)
04	ACBCID (Command ID)	ACBXLEN (ACBX length)
06	*(ACBCID continued)*	ACBXCMD (Command code)
08	ACBFNR (File number)	ACBXRSV2 (reserved 2)
0A	ACBRSP (Response code -- used for the database ID with X'30' calls)	ACBXRSP (Response code)
0C	ACBISN (ISN)	ACBXCID (Command ID)
10	ACBISL (ISN lower limit)	ACBXDBID (Database ID)
14	ACBISQ (ISN quantity)	ACBXFNR (File number)
18	ACBFBL (Format buffer length)	ACBXISNG (8-Byte ISN)
1A	ACBRBL (Record buffer length)	*(ACBXISNG continued)*
1C	ACBSBL (Search buffer length)	ACBXISN (ISN -- *included in ACBXISNG*)
1E	ACBVBL (Value buffer length)	*(ACBXISN and ACBXISNG continued)*
20	ACBIBL (ISN buffer length)	ACBXISLG (8-Byte ISN Lower Limit)
22	ACBCOP1 (Command option 1)	*(ACBXISLG continued)*
23	ACBCOP2 (Command option 2)	*(ACBXISLG continued)*
24	ACBADD1 (Additions 1)	ACBXISL (ISN lower limit -- *included in ACBXISLG*)
28	*(ACBADD1 continued)*	ACBXISQG (8-Byte ISN Quantity)
2C	ACBADD2 (Additions 2)	ACBXISQ (ISN quantity -- *included in ACBXISQG*)
30	ACBADD3 (Additions 3)	ACBXCOP1 (Command option 1)
31	*(ACBADD3 continued)*	ACBXCOP2 (Command option 2)
32	*(ACBADD3 continued)*	ACBXCOP3 (Command option 3)
33	*(ACBADD3 continued)*	ACBXCOP4 (Command option 4)
34	*(ACBADD3 continued)*	ACBXCOP5 (Command option 5)
35	*(ACBADD3 continued)*	ACBXCOP6 (Command option 6)
36	*(ACBADD3 continued)*	ACBXCOP7 (Command option 7)
37	*(ACBADD3 continued)*	ACBXCOP8 (Command option 8)
38	ACBADD4 (Additions 4)	ACBXADD1 (Additions 1)
40	ACBADD5 (Additions 5)	ACBXADD2 (Additions 2)
44	*(ACBADD5 continued)*	ACBXADD3 (Additions 3)
48	ACBCMDT (Command time)	*(ACBXADD3 continued)*
4C	ACBUSER (User area)	ACBXADD4 (Additions 4)
54	---	ACBXADD5 (Additions 5)
5C	---	ACBXADD6 (Additions 6)

Offset	ACB DSECT Field Name	ACBX DSECT Field Name
64	---	ACBXRSV3 (reserved 3)
68	---	ACBXERRG (Error offset in buffer, 64-bit -- this is not yet supported).
6C	---	ACBXERRA (Error offset in buffer, 32-bit)
70	---	ACBXERRB (Error character field)
72	---	ACBXERRC (Error subcode)
74	---	ACBXERRD (Error buffer ID)
75	---	ACBXERRE (Error buffer sequence number)
78	---	ACBXSUBR (Subcomponent response code)
7A	---	ACBXSUBS (Subcomponent response subcode)
7C	---	ACBXSUBT (Subcomponent error text)
80	---	ACBXLCMP (Compressed record length)
88	---	ACBXLDEC (Decompressed record length)
90	---	ACBXCMDT (Command time)
98	---	ACBXUSER (User area)
A8	---	ACBXRSV4 (reserved 4)

Logging the Control Blocks

All command logs (CLOGs) are written in a format as described in *Command Log Formats*.

For the sake of compatibility, the ADARUN CLOGLAYOUT (see *CLOGLAYOUT : Command Logging Format* in *Adabas Operations Manual* parameter still exists, but logs are only written in CLOGLAYOUT=8 format.

IV Adabas Buffer Descriptions (ABDs)

7 Adabas Buffer Descriptions (ABDs)

If an Adabas call using the **ACBX interface** is made that requires buffer specifications, Adabas buffer descriptions (ABDs) *must* be used. ABDs *must not* be used when specifying an Adabas call using the classic **ACB interface**; if an Adabas call using the **ACB interface** is made that requires buffer specifications, specify the buffers or pointers to the buffers directly in the Adabas call itself. For more information about the ACBX and ACB interface direct calls, read ***Calling Adabas***, elsewhere in this guide.

Adabas 8, through its ACBX interface, supports segmented buffers (multiple pairs of format and record buffers, or multiple triplets of format, record, and multifetch buffers). You can specify up to 65,535 instances of each buffer type in an ACBX call.

In mainframe system applications, the addresses of ABDs are specified directly in the Adabas call; in open system applications, the addresses of ABDs are specified in the **ABD list** associated with the call.

This chapter describes the structure of an ABD and ABD lists. For information on the defining the buffers themselves, read ***Defining Buffers***, elsewhere in this guide.

Available ABD Types

Using ABDs in an ACBX interface direct call, the buffers used in a direct call can be contiguous or discontiguous. With Adabas 8, you can define ABDs for eight different types of buffers:

- Format buffers
- Record buffers
- Multifetch buffers
- Search buffers
- Value buffers
- ISN buffers
- User buffers
- Performance buffers (reserved for use by Adabas Review only).

Each Adabas buffer segment is represented by a single ABD, although you can define multiple ABDs of a given type in the same program. Offset 4 (ABDXID) in each ABD identifies the type of buffer defined by the ABD.

In an **ACBX interface call**, there is a one-to-one correspondence between ABD and buffer specifications; each buffer you want to specify must have a corresponding ABD. The buffer can be specified in the ABD itself or referenced by indirect reference.

ABDs can be specified in any sequence in an **ACBX interface direct call**. However, if an ABD requires a matching ABD of another type, Adabas will match them sequentially. For example, if three format buffer ABDs and three record buffer ABDs are included in the call, the first format buffer ABD in the call is matched with the first record buffer ABD in the call, the second format buffer ABD is matched with the second record buffer ABD, and the third format buffer ABD is matched with third record buffer ABD.

If unequal numbers of match-requiring ABDs are specified, Adabas will generate a dummy ABD (with a buffer length of zero) for the missing ABD. For example, if three format buffer ABDs are specified, but only two record buffer ABDs are specified, a dummy record buffer ABD is created for use with the third format buffer ABD. If you would prefer that the dummy record buffer ABD be used for the second format buffer ABD instead, you must specify the dummy record buffer ABD yourself prior to the record buffer ABD to be used by the third format buffer ABD.

For commands where data in the record buffer is *not* described by a format specification in the format buffer, no format buffer segments need be specified; if any are specified, they are ignored. This applies to only a few commands; the most prominent of them is OP.

For information about the relationships between different buffer types, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

ABD Structure

The following table describes the structure of the ABD.

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
ABDXLEN	ABD length	1-2	00	2	binary
ABDXVER	Version indicator	3-4	02	2	alphanumeric
ABDXID	Buffer Type ID	5	04	1	alphanumeric
ABDXRSV1	Reserved 1	6	05	1	binary
ABDXLOC	Buffer location flag	7	06	1	alphanumeric. although a binary zero (x'00') is tolerated instead of a blank.
ABDXRSV2	Reserved 2	8	07	1	binary
ABDXRSV3	Reserved 3	9	08	4	binary
ABDXALET	ALET for buffer (if ABDXLOC=C'D')	13	0C	4	binary
ABDXSIZE	Buffer size (allocated length)	17-24	10	8	binary
ABDXSEND	Data length to send from client to the nucleus	25-32	18	8	binary

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
ABDXRECV	Data length received by the client from the nucleus	33-40	20	8	binary
ABDXADRG	64-bit indirect address pointer (if ABDXLOC=C'I' or C'D')	41-48	28	8	binary
ABDXADR	34-bit indirect address pointer (if ABDXLOC=C'I' or C'D')	45-48	2C	4	binary
---	Buffer (if ABDXLOC=C' ' or X'00')	49-n	30	user-defined	not applicable

ABD Field Descriptions

Each of the fields in the ABD is described in this section, in the order they appear in the **ABD structure**.

ABD Length (ABDXLEN)

Required. Use this field to specify the length of the ABD. Currently, the value of this field must be 48.

Version Indicator (ABDXVER)

Required. This field identifies the version of the ABD structure. A value of C'G2' in this field indicates that the buffer definition is in the new, extended ABD structure.

Buffer Type ID (ABDXID)

Required. Use this field to identify the type of buffer described by the ABD, as shown in the following table:

ID Setting	Type of Buffer
C'F'	Format
C'I'	ISN
C'M'	Multifetch
C'P'	Performance (reserved for use by Adabas Review only)
C'R'	Record
C'S'	Search
C'U'	User

ID Setting	Type of Buffer
C'V'	Value

Reserved 1 (ABDXRSV1)

This field is reserved and must be set to binary zeros.

Buffer Location Flag (ABDXLOC)

Required. Use this field to identify whether the location of the buffer is defined at an indirect address, is qualified by an ALET, or is defined at the end of the ABD itself.

If this field is set to "I" (C'I'), Adabas assumes indirect addressing is specified and will use the address specified in the indirect address pointer field (ABDXADR or ABDXADRG). In this case the buffer must reside in 31-bit or 64-bit addressable storage, in the current primary address space.

If this field is set to "D" (C'D'), Adabas assumes an ALET has been provided in **ABDXALET** and indirect addressing is implied. Adabas will use the address specified in the indirect address pointer field (ABDXADR or ABDXADRG). In this case, the buffer must reside in 31-bit or 64-bit addressable storage, addressable using the ALET.

If this field is blank (C' ') or contains hexadecimal zeros, the buffer must immediately follow the ABD in the primary address space.

Reserved 2 (ABDXRSV2)

This field is reserved and must be set to binary zeros.

Reserved 3 (ABDXRSV3)

This field is reserved and must be set to binary zeros.

ALET (ABDXALET)

If ABDXLOC is set to "D" (C'D'), this field must contain an ALET suitable to access the buffer. The ALET must be on the dispatchable unit access list (DU-al) and may relate to a dataspace or an address space.

Some special ALET values are supported. A value of zero ("0") specifies the current primary address space (PASID) and is equivalent to specifying ABDXLOC=C'I'. A value of "1" specifies the current secondary address space (SASID) and is not allowed, resulting in response code 253 (ADARSP253), subcode 14. Under z/OS, a value of "2" specifies the home address space (HASID) and is equivalent to specifying ABDXLOC=C'I' if PASID=HASID whent the command is issued.

Buffer Size (ABDXSIZE)

Required. Use this field to specify the size of the buffer (in bytes), as it is allocated. A size of zero indicates a dummy buffer, which is treated as if it was not specified at all. More than 32 KB of data can be specified in an Adabas buffer.

Data Length to Send (ABDXSEND)

Required. Use this field to specify the length of the data (in bytes) to be sent to Adabas. A buffer is sent to Adabas only if it is an input buffer for the type of command being issued.

Data Length Received (ABDXRECV)

This field specifies the length of the data (in bytes) returned to Adabas. The Adabas router sets this value at the end of call processing. The maximum value of this field will not exceed the value set for the buffer size field (ABDXSIZE). A buffer is received from Adabas only if it is an output buffer for the type of command being issued.

64-Bit Indirect Address Pointer (ABDXADRG)

If you set the buffer location flag field (ABDXLOC) to C'I' (indirect buffer) or to C'D' (ALET-qualified buffer), specify the 64-bit address of the actual buffer in this 8-byte field.

31-Bit Indirect Address Pointer (ABDXADR)

This four-byte field is an alias for the second four bytes of ABDXADRG (the 64-bit address pointer). If you set the buffer location flag field (ABDXLOC) to C'I' (indirect buffer) or to C'D' (ALET-qualified buffer), specify the 31-bit address of the actual buffer in this field and ensure that the preceding four bytes (in ABDXADRG) are zero.

Actual Buffer

If you set the buffer location flag field (ABDXLOC) to C' ' (blanks), this field should contain the actual buffer. For complete information on defining buffers, read ***Defining Buffers***, elsewhere in this guide.

ABD DSECT

The ABD DSECT can be found in member ADABDX of the distributed Adabas SRCE library.

ABD Lists

An *ABD list* is a file containing a list of pointer references to the Adabas buffer descriptions (ABDs) used for a direct call. ABD lists are used only for open systems ACBX direct calls. In the list, one ABD pointer is required for every buffer segment that is needed for the direct call.

ABD lists can include pointers to the ABDs for eight different types of buffers: format, record, multifetch, search, value, ISN, user, and performance buffers. Multiple ABDs of the same type can be specified in an ABD list.

ABDs can be specified in the list in any sequence. However, if an ABD requires a matching ABD of another type, Adabas will match them sequentially. For example, if three format ABDs and three record ABDs are included in the list, the first format ABD in the list is matched with the first record ABD in the list, the second format ABD is matched with the second record ABD, and the third format ABD is matched with third record ABD. If unequal numbers of match-requiring ABDs are listed (for example, if three format ABDs are listed, but only two record ABDs), Adabas will generate a dummy ABD for the missing ABD (in this case a dummy record ABD will be created).

For complete information about the relationships between the different types of ABD or buffer specifications, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

V

8 Defining Buffers

If your direct calls use the *ACB direct call interface*, you can define five different types of buffers: format, record, search, value, and ISN buffers. These buffers are specified elsewhere in your application and are indirectly referenced in the ACB direct call (via pointer references).

With Adabas 8, if your direct calls use the *ACBX direct call interface*, you can define eight different types of buffer segments using *Adabas buffer descriptions (ABD)* and their associated buffer definitions: format, record, multifetch, performance, search, value, ISN, and user buffers. Each Adabas buffer segment is represented by a single ABD, although you can define multiple ABDs of some types in the same program. (For example, you can define multiple format ABDs for use by the same program.) A single buffer definition is associated with each ABD -- either indirectly by pointer reference or directly in the ABD itself. For detailed information about ABDs, including their structure, read *Adabas Buffer Descriptions (ABDs)*, elsewhere in this guide.

This chapter covers the following topics:

Performance Buffers	Describes performance buffers.

9 Understanding the Different Buffer Types

The following syntax depicts the relationships between the different types of buffers that can be specified for a direct call. It should assist you in determining which buffer specifications are dependent on the presence of others.

```
[format-buffer record-buffer... [multifetch-buffer]]...
[search-buffer value-buffer]
[ISN-buffer]
[user-buffer] ...
[performance-buffer]
```

Notes:

1. If you are specifying an ACBX interface direct call, corresponding Adabas buffer descriptions (ABDs) must also be specified. In addition, in ACBX interface direct calls when buffer specifications require the presence of other buffer specifications (for example, a format buffer requires the presence of a record buffer), Adabas pairs the buffers in the sequence in which they are specified (for example, the first specified format buffer ABD with the first specified record buffer ABD). The syntax below can assist you in determining the sequence in which the ABDs should be listed in the call or in the ABD list.
2. If you are specifying an ACB interface direct call, the multifetch, performance, and user buffers listed in this syntax do not apply. In addition, buffers must be specified in this sequence: format, record, search, value, and ISN. If an earlier buffer in the sequence is not needed, but a later one is, all of the buffers up to the needed buffer must be specified, even if they are blank. For example, if an ACB interface direct call requires an ISN buffer but none of the other buffers, dummy format, record, search, and value buffers must be specified before the ISN buffer.

The following table describes the elements in this syntax:

Element	Description	Conditions
`format-buffer`	A format buffer segment to use for the call. Each format buffer segment must end with a period and be a complete and valid standalone format buffer.	Required only if you need to specify the fields to be processed during the execution of an Adabas read or update command. When required, multiple format buffers can be specified for an ACBX interface direct call. Only one format buffer can be specified in an ACB interface direct call. If a format buffer is specified in the call, a corresponding record buffer must also be specified. In an ACBX interface direct call, if a record buffer is not provided, Adabas will create a dummy one (with length zero) to pair with the format buffer. In an ACB interface direct call, if a record buffer is not provided, processing errors will occur. Optionally, in an ACBX interface direct call, a corresponding multifetch buffer can also be specified.
`ISN-buffer`	An ISN buffer segment to use for the call.	Required only if you need to set aside an area in storage to store ISNs or (in the case of an ACB interface direct call) an area to store the record descriptor elements (RDEs) of multifetched or prefetched records. When required, only one ISN buffer should be specified for the call.
`multifetch-buffer`	A multifetch buffer segment to use for the ACBX interface direct call. This buffer is only available for ACBX interface direct calls.	Used only by ACBX interface direct calls and required only if you need to set aside an area in storage to store the record descriptor elements (RDEs) of multifetched records. When required, multiple multifetch buffers can be specified for an ACBX interface direct call. If a multifetch buffer is specified, corresponding format and record buffers must also be specified. If they are not, Adabas will create dummy format and record buffers (with length zero) to correspond with the multifetch buffer.
`performance-buffer`	A performance buffer to use for the ACBX interface direct call. This buffer is only available for ACBX interface direct calls and is only used by Adabas Review.	Not required. Used only by ACBX interface direct calls withAdabas Review. For more information, read the Adabas Review documentation.

Element	Description	Conditions
`record-buffer`	A record buffer segment to use for the call.	Required only if you need to set aside an area of storage to store record data required or collected for the call. When required, multiple record buffers can be specified for an ACBX interface direct call. Only one record buffer can be specified in an ACB interface direct call. If a record buffer is specified in the call, a corresponding format buffer must also be specified. In an ACBX interface direct call, if a format buffer is not provided, Adabas will create a dummy one (with length zero) to pair with the record buffer. In an ACB interface direct call, if a format buffer is not provided, processing errors will occur. Optionally, in an ACBX interface direct call, a corresponding multifetch buffer can also be specified.
`search-buffer`	A search buffer segment to use for the call.	Required only if search criteria are required to select records for the call. If a search buffer is specified in the call, a corresponding value buffer must also be specified. Only one search and value buffer pair can be specified in a single direct call.
`user-buffer`	A user buffer segment (extension) to use for the call. The user buffer extension (UBX) is used for the user data passed to Adabas nucleus user exits 11 and 4 and Adalink user exits 1 and 2 (user exits A and B in Adabas 7).	Used only by ACBX interface direct calls and required only if the call requires input for the Adabas nucleus user exits 11 and 4 and the Adalink user exits 1 and 2 (user exits A and B in Adabas 7). You can specify a single user buffer in a direct call.
`value-buffer`	A value buffer segment to use for the call.	Required only if search criteria are required to select records for the call. If a value buffer is specified in the call, a corresponding search buffer must also be specified. Only one search and value buffer pair can be specified in a single direct call.

10 Format Buffers

Format buffers specify the fields to be processed during the execution of an Adabas read or update command. The format buffer must be long enough to hold the largest field definition contained in the related program, including the ending point (.) of the buffer itself.

Format buffers are used in combination with record buffers and, in **ACBX interface direct calls**, multifetch buffers. If a format buffer is specified, a record buffer must also be specified. If a record buffer is not provided, Adabas will create dummy record buffer (with length zero) to pair with the format buffer.

When using the ACBX direct call interface, multiple format buffers can be specified for an Adabas direct call.

In an **ACBX interface direct calls**, a multifetch buffer segment may also be listed. For more information, read *Record Buffers* and *Multifetch Buffers*, elsewhere in this guide.

For complete information about the relationships between the different types of ABD or buffer specifications, read *Understanding the Different Buffer Types*, elsewhere in this guide.

Format Buffer Syntax

Format buffers are divided into two parts: optional field selection criteria for the buffer and the record format expected.

Multiple record formats may be specified in the format buffer for files that contain various record formats. The specific record format to be used is determined by the value of a field in the file (for example, record type). When specifying multiple record formats, each one must be preceded by a format selection criterion.

The format buffer has the following syntax:

```
[field-selection-criteria1] record-format1[,[field-selection-criteria2] record-format2]... ↩
.
```

A comma must be used to separate all format buffer entries. One or more spaces may be present between entries. The last entry may not be followed by a comma.

The format buffer must end with a period.

The following sections describe the components of this syntax:

- *Field Selection Criteria*
- *Record Format Specifications*

Field Selection Criteria

Field selection criteria (*field-selection-criteria1* and *field-selection-criteria2*) are optional in a format buffer. They allow you to restrict record formats to specific values of fields. The syntax of field selection criteria is:

```
(field-name operator value1 [, value2]...)
```

field-name

The field name used in field selection criteria must be the valid name of a field in the FDT of the Adabas file being read. It *cannot* be:

- The name of a group or periodic group
- A field using any of the MU, PE, LA, or LB options
- A subfield, superfield, subdescriptor, or superdescriptor
- A collation descriptor
- A hyperdescriptor
- A phonetic descriptor.

In addition, fields specified with the NU or NC/NN options must have a non-null value; otherwise the selection criteria will be false.

operator

The following table lists the operators that can be used in the format buffer field selection criteria.

Operator	Meaning
EQ	equal to
=	equal to
NE	not equal to
LT	less than
<	less than
GT	greater than
>	greater than
LE	less than or equal to
GE	greater than or equal to

value1, value2

The value must be a numeric integer or an alphanumeric value.

If you use the EQ or = operator, a series of values can be specified, separated by commas. An alphanumeric value must be enclosed within apostrophes (for example, `'value'`).

Consider the following example:

```
(SA = 1) record-format-1, (SA = 2,3,4) record-format-2, (SA GE 4) record-format-3.
```

The field selection criteria specifies that:

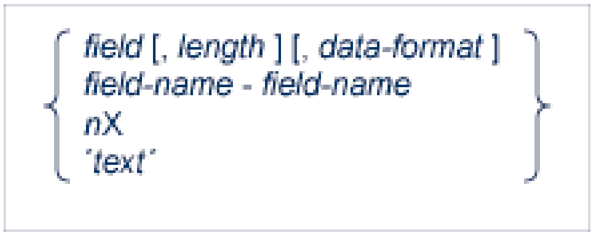
- If the value of field SA is "1", record-format-1 is used;
- If the value of field SA is "2", "3" or "4", record-format-2 is used;
- If the value of field SA is equal to or greater than "4", record-format-3 is used.

The first criterion that is met is used. If no criteria are met, a response code is returned. In the example above, if the value of field SA is "4", both the last two conditions in the field selection criteria are satisfied. However, record-format-2 will always be used, rather than record-format 3 because it is the first criteria satisfied. Likewise, if the value of SA is "0", a response code is returned.

Record Format Specifications

The record format (*record-format1*) is required and is used to indicate which fields should be read or updated in the Adabas direct call. Additional record formats can also be specified (*record-format2*).

The syntax of the record format is:



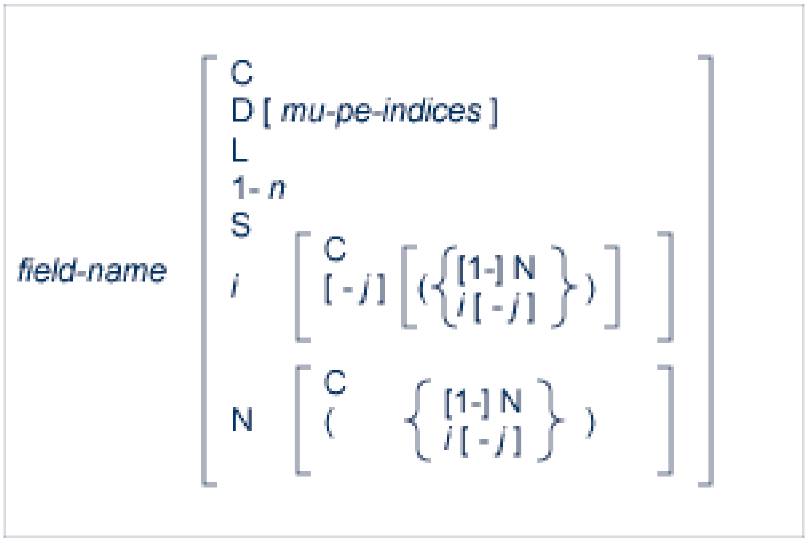
```
{ field [, length ] [, data-format ] }
{ field-name - field-name           }
{ nX                                }
{ 'text'                            }
```

For information about each of the elements in this syntax, read the section listed in the following table:

Syntax Element	Read
field	*field* *Syntax*
length	*Length and Data Format*
data-format	*Length and Data Format*
field-name - *field-name*	*Field Series Notation*
*n*X	*Space Notation (nX)*
'*text*'	*Text Insertion Notation*

field Syntax

The syntax for a `field` in record format syntax is:



This section covers the following topics related to this field syntax:

- field-name Specifications
- Index or Range Notation (i [-j] Notation)
- Periodic Group References
- Multiple-Value Fields
- Multiple-Value Fields within Periodic Groups
- Summary of Valid MU and PE Index Specifications in Format and Search Buffers
- Count Indicator (C)
- Daylight Savings Indicator (D)
- Length Indicator (L)
- Highest Occurrence/Value Indicator (N)
- SQL Significance Indicator (S)
- Selecting LOB Values or LOB Value Segments

field-name Specifications

The `field-name` is the name of the field or group for which the value, range, or count is requested or for which a new value is being provided. The name specified must be two characters in length and must be present in the FDT of the file being read or updated by the Adabas direct call. The name can refer to an elementary, subfield, superfield, multiple-value field, a group, or a periodic group.

A field name that refers to a group results in all the fields within the group being referenced. Use of group names can greatly reduce the time required to process the command. A group name cannot be used if the group contains a multiple-value or variable-length field (no standard length).

For access commands, the same name may be specified more than once. In this case, the field value is returned multiple times.

For update commands, the same name cannot be used more than once (except in the case of multiple-value fields, as explained later in this section).

A subfield, superfield, subdescriptor, or superdescriptor name may be specified for access commands but not for update commands.

The following table illustrates the relationship between allowed single-value field types and the type of command:

Command Type	Field	Subfield	Superfield	DE	SUBDE	SUPERDE	COLDE
Read (L*n*) [1]	yes	yes	yes	yes	yes	yes	yes [3]
Add (N*n*)	yes	no	no	yes	no	no	no
Find (S*n*)	yes	yes	yes	yes	yes	yes	yes
Update (A1/4) [2]	yes	no	no	yes	no	no	no

Notes:

1. Format C may be used with read commands to request the record in its compressed format.
2. A field specified for an update command cannot be repeated, cannot contain a "1-N"-type specification, and cannot specify a group and an element (field or group) contained in the group.
3. A collation descriptor (COLDE) can only be specified in the format buffer of the L9 command and only when the decode option has been specified in the user exit. The value returned is not the index value but the original field value.

Index or Range Notation (i [-j] Notation)

The following is the field name syntax for selecting multiple-value fields or occurrences of periodic groups:

```
field-name i[-j]
```

where:

i	is the periodic group or multiple-value occurrence
i-j	is the periodic group or multiple-value occurrence range

Periodic group names *must* be followed by a numeric or other appropriate suffix (see the discussions of the ***Count Indicator (C)*** and the highest occurrence/value indicator ***Highest Occurrence/Value Indicator (N)*** for more information). Specifying a periodic group name as the field name alone is incorrect syntax.

Multiple-value fields can be specified by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. See *Multiple-Value Fields* for more information.

Example	Selects . . .
ZZ3.	the third value of multiple-value field ZZ, the third occurrence of periodic group ZZ, or the first occurrence of the single- or multiple-value field ZZ contained within the third occurrence of a periodic group.
ZZ3-6.	the third through sixth values of the multiple-value field ZZ, the third through sixth occurrences of periodic group ZZ, or the first occurrence of the (single or multiple-value) field ZZ contained within the third through sixth occurrences of a periodic group.

Periodic Group References

If a periodic group (or a field within a periodic group) is to be referenced, the specific occurrence to be used must be specified. This is accomplished by appending a one- to three-digit index (value 1-191, leading zeros are permitted) to the name.

Consider the following examples:

Note: The examples in this section use the Adabas file definitions described in *File Definitions Used in Examples*.

1. An occurrence is identified by the name of the periodic group and the occurrence number.

Example	Selects . . .
GB3.	the third occurrence of periodic group GB (fields BA3, BB3, BC3).

2. When selecting fields within one or more periodic groups, the field name and occurrences (values) are used; the group name is implied.

Example	Selects . . .
BB06.	the sixth occurrence of field BB.

3. To refer to a range of occurrences of a periodic group (or a field within a periodic group), enter the periodic group or field name, followed by the first and last occurrence numbers separated by a hyphen. The occurrence numbers must occur in ascending sequence; a descending range is not permitted.

Example	Selects . . .
GB2-4.	the second through fourth occurrences of periodic group GB, including all fields within these occurrences (BA2, BB2, BC2; BA3, BB3, BC3; and BA4, BB4, BC4).
BA2-4,BC2-4.	the second through fourth occurrences of field BA and BC (BA2, BA3, BA4, BC2, BC3, BC4).

Multiple-Value Fields

Multiple-value fields can be specified in the format buffer in two ways: by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. These two methods apply as well to sub- or superdescriptors derived from multiple-value fields.

Note: The examples in this section use the Adabas file definitions described in ***File Definitions Used in Examples***.

1. Indexing: To refer to a particular value of a multiple-value field, enter a one- to three-digit index (value 1-191, leading zeros permitted) after the field name.

Example	Selects . . .
MF2.	the second value of the multiple-value field MF.
MF1, MF10.	the first and tenth values of the multiple-value field MF.

To refer to a range of values for a multiple-value field, specify the first and last values desired, connected by a hyphen, immediately after field name. An ascending range must be specified.

Example	Selects . . .
MF1-3.	the first three values of the multiple-value field MF.

2. Sequencing: To refer to multiple-value fields by repeating the field name, first specify the first value, then the second, and so on.

Example	Selects . . .
MF,MF.	the first and second values of the multiple-value field MF.
AA,MF,AB,MF,AC,MF.	the first three values of the multiple-value field MF and the values for the fields AA, AB and AC, in the order shown.

Both methods of referencing a multiple-value field can be used in the same format buffer. If no occurrence index is specified, an index that is one higher than the last index, proceeding from left to right, is implicitly (or explicitly) assigned. If the last index was specified as "N" or "1-N", referring to the highest existing occurrence, a new field specification without an explicit index will refer to the *same* field occurrence as the last specification. See ***Highest Occurrence/Value Indicator (N)*** for more information about the highest occurrence/value indicator N/1-N/NC.

For an update command where *all* format buffer references to a multiple-value field are specified without indexes (i.e., by sequencing), only those occurrences specified will remain in the record; all other occurrences that might exist are deleted. If *any one* reference to a multiple-value field is specified with an index, then only the specified occurrences are changed; all other occurrences remain unchanged.

Multiple-Value Fields within Periodic Groups

If a multiple-value field is within a periodic group, specifying the multiple-value field name implies the periodic group name. The group name, therefore, does not have to be specified; only the group occurrence or occurrence range is required. The general syntax for selecting a multiple-value field value or value range within a periodic group occurrence or occurrence range is:

```
field-name i[-j] (i[- j])
```

where:

i	is the periodic group occurrence
i-j	is the periodic group occurrence range
(*i*)	is the multiple-value field value
(*i-j*)	is the multiple-value field value range

Consider the following examples:

Note: The examples in this section use the Adabas file definitions described in ***File Definitions Used in Examples***.

1. To refer to a multiple-value field contained within a periodic group, enter the occurrence number of the periodic group after the field name, followed by the desired multiple-value field values or range of values enclosed within parentheses.

Example	Selects . . .
CB2(5).	the fifth value of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.
CB2(1-5).	the first five values of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.

2. To refer to either the same multiple-value field or the same ranges of multiple-value fields contained within a range of periodic group occurrences, enter the range of occurrences after the field name, followed by the multiple-value field value or range of values, enclosed within parentheses.

Example	Selects . . .
CB3-5(2).	the second value of multiple-value field CB in each of the third through the fifth occurrences of the periodic group containing field CB.
CB1-2(1-4).	the first four values of the multiple-value field CB in the first occurrence of the periodic group containing field CB, followed by the first four values of CB in the second occurrence of the periodic group.

Summary of Valid MU and PE Index Specifications in Format and Search Buffers

Any of the following MU and PE index specifications are valid in format and search buffer syntax:

Specification	Description
`i`	MU index for an MU field or a PE index for a PE group.
`i-j`	Range of MU indices or PE indices.
`N`	The highest MU index for an MU field or PE index for a PE group. For more information about the N indicator, read ***Highest Occurrence/Value Indicator (N)***, elsewhere in this section.
`1-N`	Range of all MU or PE indices. This specification is not allowed for update commands. For more information about the N indicator, read ***Highest Occurrence/Value Indicator (N)***, elsewhere in this section.
`i(m)`	The MU index (*m*) for an MU field in the PE group identified by the PE index (*i*).
`i(m-n)`	A range of MU indices (*m*-*n*) for an MU field in the PE group identified by the PE index (*i*).
`i(N)`	The highest MU index (**N**) for an MU field in the PE group identified by the PE index (*i*).
`i(1-N)`	All MU indices (1-**N**) for an MU field in the PE group identified by the PE index (*i*).This specification is not allowed for update commands.
`N(m)`	The MU index (*m*) for an MU field in the PE group identified by the highest PE index (**N**).
`N(m-n)`	The range of MU indices (*m*-*n*) for an MU field in the PE group identified by the highest PE index (**N**). This specification is not allowed for update commands.
`N(N)`	The highest MU index (**N**) for an MU field in the PE group identified by the highest PE index (**N**).
`N(1-N)`	All MU indices (1-**N**) for an MU field in the PE group identified by the highest PE index (**N**). This specification is not allowed for update commands.
`i-j(m)`	The MU index (*m*) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).
`i-j(m-n)`	The range of MU indices (*m*-*n*) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).
`i-j(N)`	The higheset MU index (**N**) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).

Specification	Description
`i-j(1-N)`	All MU indices (1-**N**) for an MU field in the PE group identified by the range of PE indices (*i-j*). This specification is not allowed for update commands.

Count Indicator (C)

To obtain the count of periodic group occurrences, or the count of existing values of a multiple-value field not in a periodic group, specify the periodic group or multiple-value field name followed by "C":

```
field-name C     count for multiple-value field or periodic group "field-name"
```

Example	Selects . . .
GBC	the highest occurrence number (also the occurrence count) in periodic group GB.
MFC	the number of existing values in multiple-value field MF that are not contained in a periodic group.

The count is returned in the record buffer as a one-byte binary number unless an explicit length and/or format is specified (see ***Length and Data Format***).

If your Adabas file uses the Adabas 8 extended MU/PE limits, verify that your program reads the occurrence count into a record buffer field with two or more bytes so that it can handle two-byte occurrence count values (for example, FB='MUC,2,B.'). Adabas returns response code 55 (ADARSP055), subcode 9 if you only provide a one-byte field in the record buffer for the occurrence count of an MU field or PE group in a file with extended MU/PE limits.

To obtain the count of the existing values of a multiple-value field contained within a periodic group, enter the multiple-value field name followed by the group occurrence and "C":

```
field-name i C   count for multiple-value field "field-name" within a group occurrence
```

Example	Selects . . .
CB4C.	the number of existing values of multiple-value field CB in the fourth occurrence of a periodic group.

For update commands, specifying "C" causes Adabas to skip in the record buffer the number of positions occupied by the count field, thus ignoring the count.

The user cannot directly update multiple-value or periodic group count fields. These count fields are updated by Adabas when multiple-value field values and periodic group occurrences are added or deleted.

Daylight Savings Indicator (D)

If a user session time zone uses daylight savings time, the user's local time requires a daylight savings indicator to ensure that the accurate time is reported during the hour when time is turned back from daylight savings time to standard time. Specifying the daylight savings indicator (D) on a date-time field can be used to provide you with information about whether the field's date-time value is in standard or daylight savings time.

The syntax of the daylight savings indicator is simply the letter "D" specified after the field name and before any optional MU index, PE index, or MU-PE index combinations:

```
fnD[mu-pe-index]
```

The daylight savings indicator can only be specified for date-time fields defined with the TZ option. When a daylight savings indicator is specified for a field in a format buffer, it must have a format of 2,F. The offset from standard time (in seconds) is specified in the record buffer associated with the format buffer in a halfword:

```
H'nnnn'
```

The value *nnnn* represents the number of seconds that the stored time should be offset from standard time to calculate daylight savings time. A value of zero indicates that standard time should be used; any value other than zero indicates that daylight savings time should be adjusted for and specifies the offset for that adjustment.

It is possible that the daylight savings time offset is negative if the previous standard time belongs to a different time zone and the regrouping coincided with the change to daylight savings time. In addition, any security-by-value criteria on a date-time field defined with the TZ option will be evaluated as given in UTC time.

MU and PE Index Specifications in Daylight Savings Indicator Syntax

Valid **MU and PE index specifications** in daylight savings indicator syntax can be any of the following:

- An MU index for an MU field
- A PE index for a PE group
- A range of MU indices
- A range of PE indices
- A PE index or range of PE indices and an MU index or range of MU indices for an MU field within a PE group.

For example, assuming field AA is a date-time field (or a PE group containing a date-time field), any of the following daylight savings indicator specifications might be valid:

- AAD. -- Daylight savings time adjustments should be applied to field AA.
- AAD1. --Daylight savings time adjustments should be applied to occurrence 1 of MU field AA or PE field AA.
- AAD2(5). -- Daylight savings time adjustments should be applied to occurrence 5 of the multiple-value field AA in the second occurrence of the periodic group including field AA.

Daylight Savings Indicator Rules

The following rules apply to the use of a daylight savings indicator:

1. Date-time edit masks and the daylight savings indicator are not allowed in the **field selection criteria** of a format buffer.
2. Fields with the daylight savings indicator specified in the format buffer must have formats of 2,F.
3. The daylight savings indicator can be specified only for fields defined with the TZ option.
4. Each date-time field with a daylight savings indicator in a format buffer must also be specified alone (without the daylight savings indicator) in the format buffer. For example:

   ```
   "AA,14,U,AAD,2,F."
   ```

5. If the field for which a daylight savings indicator is specified is a multiple value field, the older MU syntax is not supported. For example, instead of specifying the MU syntax "AAD,AAD,AAD.", you must specify "AAD1-3.".
6. The daylight savings indicator cannot be specified more than once for the same field, MU occurrence or MU start occurrence.
7. The following rules apply in update operations::
   - The daylight savings indicator is evaluated for times where the daylight savings time is set back to disambiguate between date-time values and when advancing the clock for daylight savings time to detect invalid nonexistent local times.
   - If the daylight savings indicator is omitted, Adabas assumes that standard time is used in cases whether the date-time value might be ambiguous.
8. In search buffers, the following rules apply:
   - A daylight savings indicator cannot be specified without a corresponding date-time field. The date-time field must be defined with one of the following date-time edit masks: DATETIME, TIMESTAMP, or NATTIME).
   - The daylight savings indicator must precede the corresponding date-time field. For example:

```
AAD,AA,14,U,E(DATETIME),GE
```

9. When sorting or reading local time values of a field with the TZ option, it may seem that the values do not appear to be in the correct order. Only when combined with the the daylight savings indicator can the local time can be precisely represented: Suppose we have stored the following DATETIME UTC values:

   ```
   2009-11-01 07:40:00
   2009-11-01 08:30:00
   2009-11-01 09:20:00
   ```

   When reading with "TZ=America/Denver", the second value or the response appears to be not in sequence:

   ```
   2009-11-01 01:40:00
   2009-11-01 01:30:00
   2009-11-01 02:20:00
   ```

   Yet the D indicator for the 3 values returns:

   ```
   H'3600'
   H'0'
   H'0'
   ```

   The first time value was when the 1 hour of daylight saving time was active.

Daylight Savings Indicator Example

Suppose the definition of field AA in the FDT is:

```
"1,AA,14,U,TZ,DT=E(DATETIME)"
```

A valid format buffer might be:

```
"AA,14,U,AAD,2,F."
```

The corresponding record buffer might be:

```
"20080814120000",H'3600'
```

In this example the daylight savings offset for field AA is 1 hour (3600 seconds) from standard time.

Length Indicator (L)

The format buffer indicator, *L*, can be used to retrieve or specify the actual length of any LA or LB alphanumeric or wide-character field value. This format buffer element is referred to as the *length indicator*.

Note: At this time, the length indicator can only be used in format buffer specifications for LA or LB fields. Support for use of the length indicator in other fields as well will be considered in a future release of Adabas.

The length indicator is specified using the field name followed by the character *L* (for example, `FB='ACL,4,B.'` would return the length of the AC field). If the field is a multiple-value field or is located in a periodic group, the field name and character L are followed by the related occurrence indices (for example, `FB='ACL2,4,B.'` would return the length of the second value of multiple-value field AC). The compressed field length is returned in four-byte binary format. A different length and format cannot be specified.

- Using the Length Indicator with MU/PE Fields
- Using the Length Indicator in Read Commands
- Using the Length Indicator in Update Commands

Using the Length Indicator with MU/PE Fields

When used with MU or PE fields, the length indicator must specify occurrence indices, including the range of occurrence index. However, it cannot specify the 1-N occurrence index. Consider the following examples.

1. In the following example, the length of the fifth value of the second occurrence of periodic group field AC would be returned:

   ```
   FB='ACL2(5).'
   ```

2. In the following example, the lengths of the first ten values of multiple value field AC would be returned:

   ```
   FB='ACL1-10.'
   ```

3. The following example is illegal as the 1-N notation is notation is not allowed with the length indicator in MU/PE fields:

```
FB='ACL1-N.'
```

4. The following example is also illegal as the length indicator does not support MU fields with out an occurrence index:

```
FB='MCL.'
```

In addition, if you elect to combine the length indicator and an **asterisk length notation** value request in the same format buffer for an MU or PE field, the value requests must use corresponding ranges as the length requests. It does not matter whether the length requests and value requests are specified in the same or different format buffer segments. Consider the following examples, where XX is an LA or LB field with the MU option:

1. The following valid examples request the length of the first two values of the XX field as well as their actual values.

```
FB='XXL1-2,XX1-2,*.'
```

```
FB='XXL1,XXL2,XX1,*,XX2,*.'
```

2. The following *invalid* examples are attempts to request the length of the first two values of the XX field as well as their actual values. However, these examples are invalid because the ranges specified for the MU field in the length and value requests are not specified in a corresponding manner.

```
FB='XXL1,XXL2,XX1-2,*.'
```

```
FB='XXL1-2,XX1,*,XX2,*.'
```

3. The following two format buffers request the length of the third and fourth values of the XX field, as well as their actual values.

```
FB='XXL3,XXL4.'
```

```
FB='XX3,*,XX4,*.'
```

4. The following invalid format buffers attempt to request the length of the third and fourth values of the XX field, as well as their actual values, but fail because the ranges specified for the length and value requests are not specified in a corresponding manner.

```
FB='XXL3,XXL4.'
```

```
FB='XX3-4,*.'
```

Using the Length Indicator in Read Commands

When the length indicator is specified for a field in the format buffer of a read command, the number of bytes required for the field value in the record buffer (without padding and with no further length indication) is returned at the corresponding field position in the record buffer. The amount of space required in the record buffer is based on the field format and the UES-related definitions for the database, file, and user.

You cannot, in the same format buffer, specify the length indicator to retrieve the length of an LA or LB field in combination with a request for the actual field value in a different format (converted) from the field's base format. For example, if character LB field L1 is stored in format A, `FB='L1L,4,B,L1,*,W.'` is an illegal format buffer specification because it requests the length of the the L1 field in addition to the value of the L1 field converted to Unicode (format W). The reason for this restriction is that the length element gives the length of the field in its native format, but the length of the value returned in the requested format (Unicode) would be different due to the conversion.

If character LB field L1 (format A) contains a 40,000-byte EBCDIC value, consider the following examples:

1. Suppose the format buffer specification for L1 is:

   ```
   FB='L1L,4,B.'
   ```

   The record buffer will contain the four-byte binary length of the value of field L1:

   ```
   X'00009C40'
   ```

2. Suppose the format buffer specification for L1 is:

   ```
   FB='L1L,4,B,L1,*,A.'
   ```

   The record buffer will contain the four-byte binary length of the value of field L1 at the beginning of the record buffer area , followed by 40,000 characters of the actual L1 data.

Empty Values, Length Indicators, and the NB (No Blank Compression) Option

If the length indicator of a field is specified in the format buffer (for example, `FB='L1L,4,B.'`), and if the field is *not* subject to blank compression (the NB option is specified for the field in the FDT), the length returned is the number of bytes specified when the value was stored (which can be zero). However, if the field is subject to blank compression, the length returned is the number of significant left-most bytes, beyond which the value is padded with blanks. For an all blank value, the returned length is the length of one blank (one byte for alphanumeric fields, two bytes for wide-character fields).

The following examples depict this. For the first three examples, suppose the FDT is defined as follows:

```
1 | Z1 | 022 | A | NU
1 | Z2 | 000 | A | LB NU
1 | Z3 | 022 | A | NU
```

Note: In Natural, Z2 is a large object (LOB) field with DYNAMIC specified in the DDM. So note that the field length is zero ("000") in the FDT.

Example 1: A Natural application fills all three fields:

```
Z1-FIELD := '1234567'
Z2-FIELD := 'LOB LOB LOB'
Z3-FIELD := '89101112'
```

Result: Response Code 0 (ADARSP000).

Example 2: A Natural application fills fields Z1 and Z3:

```
Z1-FIELD := '1234567'
Z3-FIELD := '89101112'
```

Result: NAT3052 Error processing a buffer. DB/FNR/Subcode xxx/yyy/2. Invalid length for variable-length field specified in record buffer.

Example 2 failed because an attempt was made to store field Z2 with a length of zero. Adabas assumes that fields cannot be stored with a length of zero unless they are defined with the NB option. If a field is not defined with the NB option (as with Z2 in this case), but an attempt is made to store it with a length of zero, an error results. To resolve this, store the field you want empty as a single blank (as shown in the next example) or define it as an NB field (as shown in Example 4).

Example 3: A Natural application fills fields Z1 and Z3 with values, but fills field Z2 with only the initial value of an Alpha field (blank):

```
Z1-FIELD := '1234567'
Z2-FIELD := ' '
Z3-FIELD := '89101112'
```

Result: Response Code 0 (ADARSP000).

Now suppose the definition of field Z2 in the FDT is *not* subject to blank compression (the NB option has been added to its FDT definition):

```
1 | Z1 | 022 | A | NU
1 | Z2 | 000 | A | LB NU NB
1 | Z3 | 022 | A | NU
```

In this case, Example 2 above would respond differently, as shown below in Example 4.

Example 4: A Natural application fills fields Z1 and Z3, when Z2 includes the NB option:

```
Z1-FIELD := '1234567'
Z3-FIELD := '89101112'
```

Result: Response Code 0 (ADARSP000).

Using the Length Indicator in Update Commands

When a length indicator is specified in the format buffer for an update command, the corresponding value in the record buffer specifies the actual value length of the field in the record buffer. Only one length indicator for the base field can be specified and it must be accompanied by the **asterisk (*) length notation** in the format buffer.

The length indicator must occur in its format buffer segment prior to any format element that implies a variable length in the record buffer (due to the use of asterisk notation or zero length notation). In other words, the length indicator is located in a constant position, independent of the values of any fields mentioned in the format.

Highest Occurrence/Value Indicator (N)

The indicator "N" selects the last value in a series of values comprising a multiple-value field, or the last occurrence of a periodic group, removing the need to know the number of the last value or occurrence.

Note: The 1-N notation is not supported for LB fields.

The notation "1-N" selects all values comprising a multiple-value field, or all occurrences of a periodic group. For multiple-value fields in periodic groups, it is not possible to combine the specification 1-N for the group occurrence with any specification for the field occurrences.

The notation "NC" selects the count of the existing values of a multiple-value field in the last occurrence of the periodic group containing the field.

`field-name  N`	Last occurrence of periodic group `field-name` or the highest value of multiple-value field `field-name`.
`field-name  NC`	Count of values for multiple-value field `field-name` in the last occurrence of the periodic group containing `field-name`.
`field-name  NC`	Selects all values of multiple-value field `field-name` or all occurrences of the periodic group `field-name`.
`field-name  N (N)`	The last value of multiple-value field `field-name` in the last occurrence of the periodic group containing `field-name`.
`field-name  N(1-N)`	All values of multiple-value field `field-name` in the last occurrence of the periodic group containing `field-name`.

`field-name  N( i [- j ↩`
`] )`
	Value or range of values of multiple-value field `field-name` in the last occurrence of the periodic group containing `field-name`.
	For multiple value LB fields and LB fields within periodic groups, you must specify a specific occurrence of the field. In the following example, the first value of multiple-value LB field L2 is selected:
	`FB='L21.'`
	You cannot specify the base field without an occurrence index. For example, the following format buffer specification is *not* valid:
	`FB='L2.'`

`field-name i [- j ] ↩`
`(N)`
	The last value of multiple-value field `field-name` in an occurrence or range of occurrences of the periodic group containing `field-name`.
	For multiple value LB fields and LB fields within periodic groups, you must specify a specific occurrence of the field. In the following example, the fifth value of LB field L3 in its second PE-group instance is selected:
	`FB='L32(5).'`

`field-name  i [-j ] ↩`
`(1-N)`
	All values of multiple-value field `field-name` in an occurrence or range of occurrences of the periodic group containing `field-name`.

Example	Selects . . .
MFN.	the highest (last) value of multiple-value field MF. If multiple-value field MF contains four values, the fourth value (the last value entered) is selected.
GBN.	the highest occurrence number of a periodic group GB.
MFNC.	the count of existing values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MF1-N.	all values of the multiple-value field MF.
GB1-N.	all occurrences of periodic group GB.

Example	Selects . . .
MFN(1-N).	all values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(N).	the highest value for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(4).	the fourth value of the multiple-value field MF in the highest occurrence of the periodic group containing MF.
CB3-5(N).	the highest value of the multiple-value field CB in the third through fifth occurrences of the periodic group containing CB.
CB2(1-N).	all values of the multiple-value field CB in the second occurrence of the periodic group containing CB.

SQL Significance Indicator (S)

The "S" significance, or null, indicator and the corresponding null indicator value in the record buffer indicate whether a field's value is significant, including zero or blank, or not significant (undefined). The S indicator can only be applied to elementary fields that are defined with the NC option, but not for an NU option field:

```
field-name S    SQL significance indicator
```

See the section *Record Buffer* for a description of the null value indicator, and the ADACMP utility description in the *Adabas Utilities* documentation for information about defining SQL null fields.

The related null indicator value in the record buffer, described below, has a two-byte standard length and fixed point format; this length and format cannot be overridden.

For update-type commands, a null value will be stored in the field if the corresponding null indicator value is X`FFFF' and no NN option is specified for the field. Otherwise, the null indicator must be X`0000'. This, combined with the S indicator, shows that the field's value given elsewhere in the record buffer is to be stored.

The S indicator is not required for update-type commands; if the S indicator is not specified, the field value is updated exactly as if the S indicator had been specified and the corresponding null indicator value had been set to zero (a null in the field is a value). See the examples below.

For read-type commands, the S indicator is required when the NC fields are defined without the NN option. If the S indicator is not present when a read command detects an NC-specified field and the field actually contains a null value, a response code 55 (ADARSP055) with subcode 5 is returned.

Example:

This example uses the "field-name S" indicator with the two-byte null indicator in the record buffer to update or add a record, setting field AA equal to the SQL null value and ignoring the value for field AA:

Format Buffer	`AAS , AA , 2 , ... .`	Name of the fields to be read
Record Buffer	`FFFF  123F`	Field values returned by Adabas

For examples showing the use of the SQL significance indicator when a group or range of fields containing an NC field is specified, see the section ***The SQL Significance Indicator and Field Series Notation***.

The "field-name S" indicator can be anywhere within the format buffer; that is, it need not precede the corresponding field element. For update-type commands, the format buffer cannot contain more than one element referring to the same field:

AAS.	valid for read/update commands
AA.	valid for read/update commands
AAS,AA,AAS.	valid for read commands only

This means that several format buffer elements referring to the same field cannot be specified for an update-type command:

AA,AA.	causes response code 44 (ADARSP044)
AA,AAS,AA.	causes response code 44 (ADARSP044)

Selecting LOB Values or LOB Value Segments

Complete or partial large object (LB) fields (LOB fields) can be specified using the following syntax for `field-name`:

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

The following substitutions can be used in this LOB field syntax:

- A two-character LOB field name should be specified for `field-name`. The LOB field may be an elementary or a multiple-value LOB field and can be located in a periodic group. This is the LOB field whose whole or partial value you intend to read or write.
- If the LOB field is a multiple value field or a field in a periodic group, you must specify the appropriate `index-or-range` notation used for MU or PE fields to identify the specific MU or PE field occurrence you want to read or write. If you only want a partial LOB value (if you specify

the `(bytenum,LOBlength[,LOBlength2])` construct), a single occurrence of an MU or PE LOB field must be specified; in this case the range notation is not allowed.

- The `(bytenum,LOBlength[,LOBlength2])` construct is optional and is known as *LOB segment notation*. It can be used to select only part (a segment) of a LOB value to be read or written. If this construct is omitted, the entire LOB field value is read or written. To use this construct, the field name must defined with the LB (LOB field) option.

  You cannot specify the same LOB field (or the same instance of an MU or PE LOB field) more than once in the same format buffer (with our without LOB segment notation) in a single store or update command (N1 or A1 command).

- Valid values for `bytenum` in LOB segment notation are unsigned decimal numbers or an asterisk (*). If a number is specified, it must lie in the range from 1 through 2,147,483,647. If a number is specified, it denotes the starting position of the byte within the LOB value where the LOB segment read or write should begin. The first (leftmost) byte of an entire LOB value is byte number one (1).

  If an asterisk (*) is specified, it denotes the current position in the LOB value. The current position of a LOB value is tracked in the ISN Lower Limit (ACBISL or ACBXISL) field of the Adabas control block, which contains the number of bytes preceding the segment you now want to read or update. The asterisk notation is valid in A1 and L1/L4 commands that also have their Command Option 2 (ACBCOP2 or ACBXCOP2) fields set to "L". It is useful for reading or writing a LOB value using multiple calls with a single, constant format, as it avoids the need to adjust the `bytenum` value in the format for each call in the sequence. At the end of each of these calls, Adabas returns in the ISN Lower Limit field the byte number of the last byte of the LOB segment just processed. Only one format element can be specified that uses the asterisk positioning notation in all sections of the format buffer.

- Valid values for `LOBlength` in LOB segment notation are unsigned decimal numbers ranging from 0 to 2,147,483,647. It specifies the length, in bytes, of the LOB value segment that should be read or written. Space for a LOB value segment of this size must be provided at the corresponding position in the record buffer segment. If a numeric value is specified for `bytenum` in LOB segment notation, the sum of the byte number referenced by `bytenum` and the value of `LOBlength` must be less than or equal to 2,147,483,647.

- Valid values for `LOBlength2` in LOB segment notation are unsigned decimal numbers ranging from 0 to 2,147,483,647. `LOBlength2` is optional, and can only be specified in write operations. When it is specified, its value must equal the value of `LOBlength`. `LOBlength2` indicates that the LOB value segment in the record buffer should replace a segment of the same size in an existing LOB value, without deleting any existing remainder of the value.

- Length and format overrides are `not` allowed in format elements for LOB segments (when LOB segment notation is used in the format buffer). For example, the following notations are all *invalid*:

  - `FB = 'LB(100001,25000),0,A.'`
  - `FB = 'LB(100001,25000),*,A.'`

- `FB = 'LB(100001,25000),25000,A.'`

On the other hand, the following notations are all *valid*:

- `FB = 'LB.'`: Selects the entire LOB value.
- `FB = 'LB,100000.'`: Selects 100,000 bytes of data from the beginning of the LOB value.
- `FB = 'LB(100001,25000).'`: Selects 25,000 bytes of data starting at the 100,001st byte of the LOB value.
- `FB = 'LB(100001,25000,25000).'`: Selects (in an update) 25,000 bytes of data replacing the same number of bytes starting at the 100,001st byte of the LOB value.
- `FB = 'LB2(100001,25000).'`: Selects 25,000 bytes of data starting at the 100,001st byte of the LOB value, which is the second occurrence of a multiple-value or periodic-group LOB field.

For complete information about processing partial LOB fields, read *Processing LOB Field Segments*, in the *Adabas DBA Tasks Manual*.

Length and Data Format

The length and format parameters are used if a field value is being provided or is to be returned in a length or format different from the standard defined for the field in the FDT. If the length or format parameters are omitted, the field value must be provided or is returned in the standard length and format of the field:

```
[ , length ] [ , data-format ]
```

Possible format and length conversions are suggested by the information in the following table. A format conversion cannot be specified for subfields or subdescriptors; superfields or superdescriptors; or hyperdescriptors.

Fmt	Max Length (in bytes)	Data Type	Compatible Formats
A	253	alphanumeric, left-justified	W
W	253 [1]	wide-character, left-justified	A
A,W	16,381 [1, 2, 3]	alphanumeric or wide-character with LA (long alpha) option; left-justified; preceded by optional two-byte binary (inclusive) length	W,A
A	2,147,483,643[3, 4]	alphanumeric with LB (LOB) option; left-justified; preceded by optional four-byte binary (inclusive) length	None
B	126	binary; right-justified; unsigned	A,F,P,U
F	8	fixed-point; right-justified; signed; two, four, or eight bytes	A,B,P,U
G	8	floating-point; four or eight bytes	none

Fmt	Max Length (in bytes)	Data Type	Compatible Formats

P	15	packed decimal; signed;
		positive=A,C,E, or F; negative=B or D
			A,B,F,U

U	29	unpacked decimal; signed;
		positive=A,C,E, or F; negative=B or D
			A,B,F,P

Note:

1. Like an alphanumeric field, a wide-character field may be a standard length in bytes defined in the FDT, or variable length. Any non-variable length override for a wide-character field must be compatible with the user encoding; for example, a user encoding in Unicode requires an even length (maximum of 252 bytes for non-LA fields, maximum of 16,380 bytes for LA fields).
2. If the LA value in the record buffer is preceded by its two-byte length (variable field length notation), the maximum length value is 16,383 bytes.
3. For LA and LB fields only, you can specify an asterisk (*) instead of a length in the format element. For more information, read ***Asterisk (*) Length Notation***, elsewhere in this section.
4. If the LOB value in the record buffer is preceded by its four-byte length (variable field length notation), the maximum length value is 2,147,483,647 bytes.

The length specified must be large enough to contain the value in the chosen format, but cannot exceed the maximum length permitted.

If a length of zero is specified, or if no length is specified and *field-name* refers to a variable-length field (no standard length), the amount of record buffer space used for the field is variable and depends on its actual value. The value returned by Adabas begins with a length indicator that specifies the value's length in binary format, including the space for the indicator itself. For update commands, you must provide this length indicator at the beginning of the field value in the record buffer.

- For a field without the LA- and LB-options, the length indicator is one byte.
- For an LA field, the length indicator is two bytes.
- For an LB field, the length indicator is four bytes.

For example, if the length in the format buffer is given as zero, an LA field with a 1000-byte value is represented in the record buffer as two bytes containing the number 1002 in binary format, followed by 1000 bytes containing the field value proper.

The format specified must be compatible with the standard format of the field.

- Conversion between packed/unpacked decimal values and binary is limited to values between 0 and 2,147,483,647.
- Conversion from a numeric format to alphanumeric results in an unpacked value, left justified, without leading zeros and with trailing blanks. For example, the three-byte packed value "10043F" would be converted to "F1F0F0F4F3404040". Value truncation is possible with this type of conversion.

Note: Length and format overrides are not allowed in format elements for LOB segments (when LOB segment notation is used in the format buffer).

The following additional topics are covered in this section:

- Asterisk (*) Length Notation
- Edit Mask Notation (Read Operations Only)
- Examples

Asterisk (*) Length Notation

For LA and LB fields only, you can specify an asterisk (*) instead of a length in the format element. Asterisks cannot be specified for regular alphanumeric or wide-character fields. The presence of an asterisk indicates that the amount of space available for the LB field value in the record buffer is variable and depends on the actual value of the LB field. However, unlike the zero length specification setting, *no* four-byte length field precedes the LB field value in the record buffer; the record buffer area corresponding to the LB format element only contains the value of the LB field. The actual LB field value length *should be* retrieved for read commands and *must be* specified for update commands using the new format buffer length indicator, *L*. For more information about the length indicator, read ***Length Indicator (L)***, elsewhere in this guide.

In the following example, the record buffer for LB field L1 contains only the value of the L1 field, followed by the value of the AA field for which 10 bytes have been allotted.

```
FB='L1,*,AA,10,A.'
```

In the following example, the record buffer for LB multi-value field L2 contains the first ten values of L2.

```
FB='L21-10,*.'
```

The record buffer is not necessarily required to provide sufficient space for the entire field if its format element includes an asterisk length setting. However, in read command processing, the field value can be truncated if *both* of the following conditions are met:

- The record buffer space available is insufficient for the field value.
- A field with asterisk notation is specified at the *end* of the format buffer.

In these conditions, no error is returned. If this were the case in the second example above (`FB='L21-10,*.'`), Adabas would truncate the ten values to be read down to the length of the corresponding record buffer segment. (The truncation occurs from right to left; that is, the last value is truncated first; if the remaining space is still insufficient, the second-to-last value is truncated, and so on.) In extreme cases, if no space is available at all for the field value, the value is truncated down to zero bytes.

In the first example above (`FB='L1,*,AA,10,A.'`), if the record buffer segment is too short, no truncation occurs because this is not allowed for fields specified with a fixed length or length of zero (0). Rather, the nucleus returns response code 53 (ADARSP053), indicating that the record buffer is too small.

Only read commands executed by the Adabas nucleus may truncate values specified with the asterisk notation; no truncation occurs in update commands. In addition, the ADACMP utility does not truncate values specified with the asterisk notation.

Edit Mask Notation (Read Operations Only)

Edit masks are used according to the standard edit mask rules used in the COBOL programming language.

An edit mask may only be specified for numeric fields. All data returned by Adabas to an edited field is converted to unpacked decimal format regardless of the standard format of the field. A maximum of 15 digits (not counting edit characters) can be returned to an edited field.

For a field with an edit format specified, the length parameter must be large enough to contain the field value plus all required edit characters.

Format	Generates the edit mask . . .
E1	zzzzzzzzzzzzzzz
E2	zzzzzzzzzzzzzz9-
E3	zzzzzzzzz99.99.99
E4	zzzzzzzzz99/99/99
E5	z.zzz.zzz.zzz.zzz,zz
E6	z,zzz,zzz,zzz,zzz.zz
E7	z,zzz,zzz,zzz,zz9.99-
E8	z.zzz.zzz.zzz.zz9,99-
E9	*,***,***,***,**9.99-
E10	*.***.***.***.**9,99-
E11	user-designated mask
E12	user-designated mask
E13	user-designated mask
E14	user-designated mask
E15	user-designated mask

Note: Although edit formats E3 and E4 provide space for the century digits (see the following examples), they do not *enforce* date formats that are compatible with year 2000 requirements.

For information on date-time edit masks, supplied with Adabas, read *Date-Time Edit Mask Reference*, in the *Adabas DBA Tasks Manual*. Date-time edit masks can be used for record updates, in addition to reads. For information on how these date-time edit masks can be used and processed in format and search buffers, read *Date-Time Edit Mask Processing in Format and Search Buffers*, elsewhere in this guide.

Examples

Format Buffer	Field Value	Edited Value
XC,15,E1.	009877	bbbbbbbbbbb9877
XC,8,E4.	301177	30/11/77
XB,5,E7.	-366	3.66-
XB,7,E9.	542	**5.42
Y2,10,E4.	20000229	2000/02/29

Field Series Notation

The notation "field-name - field-name" may be used to refer to a series of consecutive fields (as ordered in an FDT). The user specifies the beginning and ending field names connected by a dash:

```
field-name - field-name  series notation
```

No multiple-value field or periodic group may be contained within the series.

A name that refers to a group may not be specified as the beginning or ending name, but a group may be embedded within the series.

Standard format and length is in effect for all the fields within the series. No length or format override is permitted.

Example	Selects . . .
AA-AC.	the fields AA, AB, and AC.
AA-GC.	nothing. The series may not contain a multiple-value field or a periodic group.
GA-AC.	nothing. The series may not begin/end with a group.
AA,5,U,-AD.	nothing. A length and/or format override is not permitted in a series notation.

The SQL Significance Indicator and Field Series Notation

When a group or range of fields contains a field specified with the NC option, the corresponding S operator is optional for read (L*x*) commands. For update (A1) commands, the S operator must not be specified. Adabas assumes that the null indicator corresponding to the NC field in the format buffer is located just in front of the field's value in the record buffer.

For example, given the following field definitions in the FDT:

```
01,GR
  02,AA,8,A
  02,BB,8,A,NC
  02,CC,8,A
```

If the format buffer of an update-type command specifies "GR." or "AA-CC." , the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
```

That is, the null indicator must be included in the record buffer sequence, although the S indicator was not (and must not be) specified in the format buffer.

If the format buffer of a read (L*x*) command specifies "GR,BBS." or "AA-CC,BBS.", the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
null-indicator-BB
```

In other words, the first appearance of the null indicator is implied in the record buffer while the second appearance was explicitly called for by the format buffer.

Space Notation (nX)

The nX syntax is used differently for read and update commands:

```
nX
```

For read commands, nX indicates that "n" spaces are to be inserted in the record buffer by Adabas immediately before the next field value:

Format Buffer	`AA , 5X , BB.`	Name of the fields to be read
Record Buffer	`value-AA 5-blanks value-BB`	Field values returned by Adabas

For update commands, nX causes "n" positions in the record buffer to be ignored by Adabas:

Format Buffer	`XX , 5X , YY.`	Name of the fields to be updated
Record Buffer	`value-XX ignore-5-bytes value-YY`	Field values provided by user

Text Insertion Notation

The text syntax is used differently for read and update commands:

```
'text'
```

For read commands, the character string specified in the format buffer is to be inserted in the record buffer immediately before the next field value. The character string provided can be 1-255 bytes long, and may contain any alphanumeric character except a quotation mark.

For example:

Format Buffer	`AA,'here is some text', BB.`	Name of the fields to be read
Record Buffer	*`value-AA`* `here is some text` *`value-BB`*	Field values returned by Adabas

For update commands, the number of positions enclosed within the apostrophes in the format buffer will be ignored in the corresponding positions of the record buffer.

Specifying Field Lengths of LA (Long Alpha) Fields in Format Buffers

The LA option is normally used with variable-length data. The length of an alphanumeric field with the LA option can also be specified in the format buffer. However, the length is then limited to 16381 bytes:

Format Buffer	`BA,5, ... .`	Name of the fields to be read
Record Buffer	`ABCDE`	Field values returned by Adabas

A field with LA option can also have the NU (null suppression), NC/NN SQL null significance, or the MU (multiple-value field) option, and can be a member of a PE (periodic) group.

A field with the LA option cannot be a descriptor and cannot be the parent of hyperdescriptors or phonetic descriptors.

A field is compressed the same way, with or without the LA option; that is, by removing trailing blanks. This must be kept in mind if you store binary data in an LA field.

Specifying Field Lengths of LOB (Large Object) Fields in Format Buffers

The LB option is normally used with variable-length data. The length of an alphanumeric field with the LB option can also be specified in the format buffer.

Format Buffer	`L1,100000, ... .`	Name of the fields to be read
Record Buffer	`ABCDE    all together 100000 bytes of data`	Field values returned by Adabas

Format Buffer Performance Considerations

Performance improvements may be achieved by using the following guidelines during format buffer construction:

- Use group names wherever possible rather than referring to elementary fields individually. Use of group names reduces the time required by Adabas to interpret the format buffer.
- Use of the field series notation does not result in performance improvements. A field series notation is converted by Adabas into a series of elementary fields.
- Use length and format overrides only when necessary. Using overrides requires additional processing time when interpreting the format buffer and when processing the field.
- If the same fields of a record are to be read and then updated, the same format buffer should be used for the read and update commands. For more information, refer to the descriptions of command and format IDs.
- You should request periodic (PE) group and multiple-value (MU) field occurrences, based on the requirements of the application. In other words, do not arbitrarily request all occurrences; doing so requires extra time to translate the format buffer, and may mean decompressing numerous empty occurrences. Generally, the AA1-N field argument is the most efficient for selecting periodic group occurrences.

11 Record Buffers

A record buffer defines an area in storage to which Adabas can return data or in which you supply data for processing. When a record buffer is required, a corresponding format buffer is expected as well. If a format buffer is not provided, Adabas will create a dummy format buffer (with length zero) to pair with the record buffer. For complete information about the relationships between the different types of ABD or buffer specifications, read *Understanding the Different Buffer Types*, elsewhere in this guide.

When using the **ACBX direct call interface**, multiple record buffers can be specified for an Adabas direct call.

Record buffers are used primarily with read, search, and update commands:

- For read commands, the values of the fields specified in the **format buffer** are returned by Adabas in the record buffer. They are returned in the order specified by the **format buffer**. Here is an example of a format buffer and its corresponding record buffer.

Format Buffer	`AA,BB.`	Name of the fields to be updated
Record Buffer	*`value-AA value-BB`*	Field values provided by user

  Each value is returned in the standard length and format defined for the field unless a length or format override was specified in the **format buffer**. If the value is a null value, it is returned in the format that is in effect for the field, as follows:

Field Type	Null value represented by . . .
Alphanumeric (A)	blanks (hex '40') or blank of user override encoding
Binary (B)	binary zeros (hex '00')
Fixed (F)	binary zeros (hex '00')
Floating Point (G)	binary zeros (hex '00')
Packed (P)	decimal packed zeros with sign (hex '00' followed by '0A', '0B', '0C', '0D' or '0F' in the rightmost, low-order byte)
Unpacked (U)	decimal unpacked zeros with sign (hex 'F0' followed by 'C0' or 'D0' in the rightmost, low-order byte)
Wide-character (W)	Unicode blanks (hex '20') or blank of user override encoding

  Note: SQL-compatible null values in NC/NN option fields require the additional null value and significance indicator. See *Specifying and Reading the SQL Null Indicator in Record Buffers*, and *SQL Significance Indicator (S)*.

  Adabas returns the number of bytes equal to the combined lengths (standard or overridden) of all requested fields.

- For add or update commands, the new values for the fields specified in the format buffer are provided by the user in the record buffer.

Format Buffer	`XX,YY.`	Name of the fields to be updated
Record Buffer	`value-XX value-YY`	Field values provided by user

When updating a record, you must specify the new value in the record buffer. If a null value is being provided, it must be provided according to the field type in effect, as described above.

- The record buffer is also used to transfer information between the user program and Adabas in the following commands:

Command	Data Provided	Data Returned
OP	Files to update and the operation type (ET, exclusive control) The user's individual time zone	User data (optional)
LF	-	Field definitions for the file
RE	-	User data stored in system file
C5	Protection log user data	-
ET/CL	User data (optional)	-

For the OP command, the record buffer indicates the type of user and the files to be used and optionally, the time zone of the user..

The record buffer is also used for user data (OP, RE, CL, ET commands).

Specifying and Reading the SQL Null Indicator in Record Buffers

To support Adabas SQL Gateway (ACE) and other structured query languages (SQLs), fields defined with the NC/NN (not-counted/null-not-allowed) options indicate an SQL-significant null with a two-byte binary null indicator in the record buffer.

Whether a field's zero value is significant or an irrelevant null (unspecified) depends on the null indicator specified in the record buffer when the value is entered or changed, or returned in the record buffer when the value is read.

In addition to specifying or reading the value itself, either:

- set the null indicator into the record buffer position that corresponds to the field's designation in the format buffer for an update operation, or
- ensure that your program examines the null indicator (if any) returned in the record buffer position corresponding to the field's position in the format buffer for a read operation.

The null indicator is always two bytes long and has fixed-point format, regardless of the data format.

For a read (L*x*) or find with read (S*x* with format buffer entry) command, the null indicator value returns one of the following (hexadecimal) null indicator values, according to the actual value that the selected field contains:

Hex Value	Description
FFFF	A null value in the field is not significant.
0000	A null value in the field is a significant value; that is, a true zero or blank.
xxxx	The field is truncated. The null indicator value contains the length (xxxx) of the entire value as stored in the database record if the length is less than 32,768.
0001	The field is significant and the value is truncated, and the length of the value does not fit into the S element because it is greater than 32, 767.

For an update (Ax) or add (Nx) command, the (hexadecimal) null indicator value in the record buffer must be set to one of the following values:

Hex Value	Description
FFFF	The field value is set to "undefined", an insignificant null; the field's contents in the record buffer are irrelevant when set to binary zero or blank characters.
0000	If either no value is specified in the record buffer, or binary zero or blanks are specified, the field contains a significant null value.

For an *add* command, if no value for the field is supplied in the record buffer for a field defined with the NC option, the field is treated as a null field. The following example shows how a null would be represented in a two-byte Adabas binary field AA defined with the NC option:

Field definition: 01,AA,2,B,NC

	For a nonzero value	For a blank	For null
Null Value indicator in Record Buffer	0 (binary value is significant)	0 (binary null is significant)	FFFF (binary null is not significant)
Data	0005	0000 (zero)	not relevant
Adabas internal representation	0205	0200	C1

For an *update* (A1/N1) command, the field value is always significant whenever the field is defined with the NC option; the field is treated as if a hexadecimal null indicator value of "0000" has been specified.

For a *read* command, if the null indicator is not specified for an NC option field, the field value is returned in the record buffer whenever there is a significant value in the record. If the Data Storage record contains a "not significant" (FFFF) indicator value for the field, response code 55 (ADARSP055) will be returned when the record is read.

Specifying Field Lengths of LA (Long Alpha) Fields in Record Buffers

The LA option is normally used with variable-length data. If the length is not explicitly specified in the format buffer then the length of an alphanumeric field with the LA option can be specified in the record buffer or will be returned by Adabas.

The field value is preceded by a two-byte length field containing the length of the value, plus 2 (inclusive length).

Format Buffer	`BA, ... .`
	or
	`BA,0, … .`
		Name of the fields to be read without explicit length specifications.

Record Buffer	`X´0005C1C1C1´...`
	or
	`X´2712...(10,000 ↩ characters)...`
		Field values specified for Adabas or returned by Adabas in hexadecimal.
		In the first case the value is three bytes long; in the second case the value is 10,000 bytes long.

Specifying Field Lengths of LOB (Large Object) Fields in Record Buffers

The LB option is normally used with variable-length data. If the length is not explicitly specified in the format buffer then the length of an alphanumeric field with the LB option can be specified in the record buffer or will be returned by Adabas.

The field value is preceded by a four-byte length field containing the length of the value, plus 4 (inclusive length).

Format Buffer	`L1, ... .`
	or
	`L1,0, … .`
		Name of the fields to be read or updated without explicit length specifications.

Record Buffer	`X´00000007C1C1C1´...`
	or
	`X´000186A4...(100,000 ↩ characters)...`
		Field values specified for Adabas or returned by Adabas in hexadecimal.
		In the first case the value is three bytes long; in the second case the value is 100,000 bytes long.

Specifying the Daylight Savings Time Offset in Record Buffers

The daylight savings offset from standard time (in seconds) is specified in the record buffer associated with the format buffer in the following format:

```
H'nnnn'
```

The value *nnnn* represents the number of seconds that the stored time should be offset from standard time to calculate daylight savings time. A value of zero indicates that standard time should be used; any value other than zero indicates that daylight savings time should be adjusted for and specifies the offset for that adjustment.

Suppose the definition of field AA in the FDT is:

```
"1,AA,14,U,TZ,DT-E(DATETIME)"
```

A valid format buffer might be:

```
"AA,14,U,AAD,2,F."
```

The corresponding record buffer might be:

```
"20080814120000",H'3600'
```

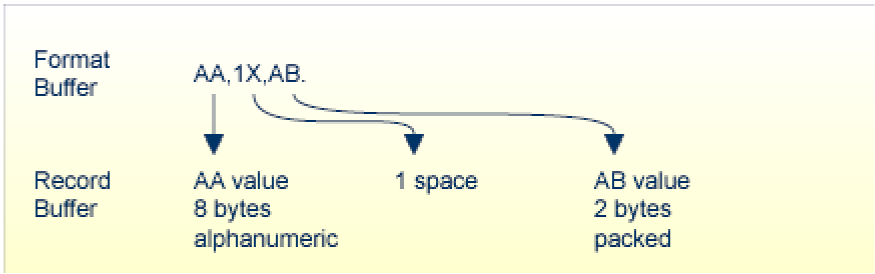
In this example the daylight savings offset for field AA is 1 hour (3600 seconds) from standard time.

Read ***Daylight Savings Indicator (D)***, in the format buffer description elsewhere in this guide for more information about daylight savings indicator usage in format and search buffers.

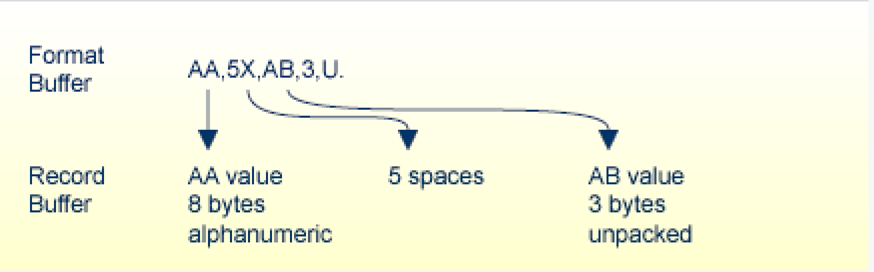
12 Format and Record Buffer Examples

This section provides examples of format and record buffer construction. For the Adabas file definitions used in all the examples in this section, see ***File Definitions Used in Examples***.

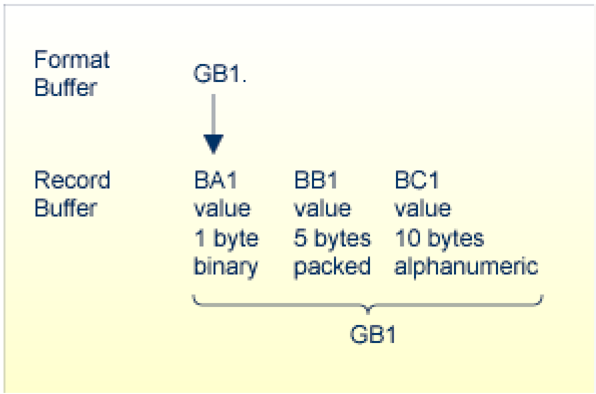
Example 1: Using Elementary Fields (Standard Length and Format)



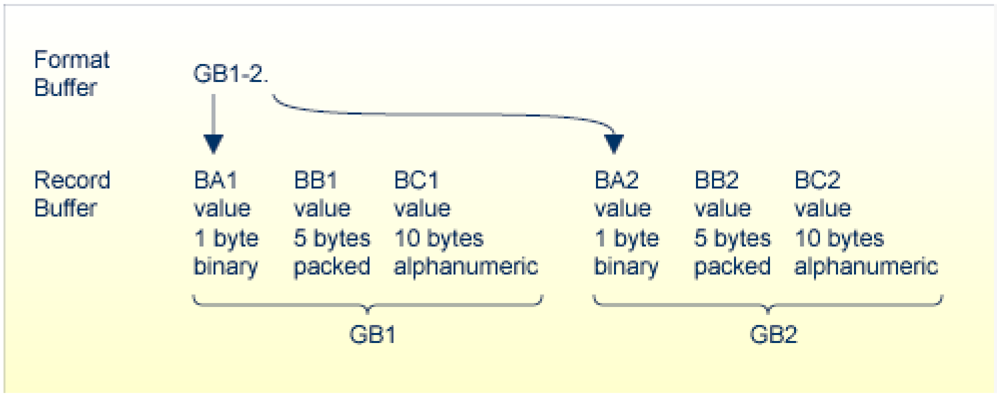
Example 2: Using Elementary Fields (Length and Format Override)



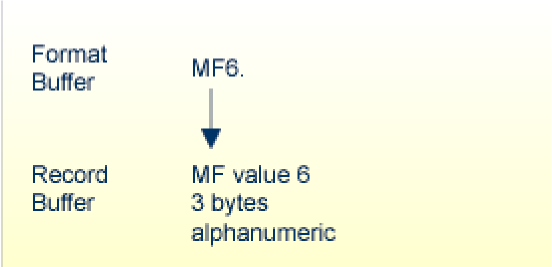
Example 3: A Reference to a Periodic Group



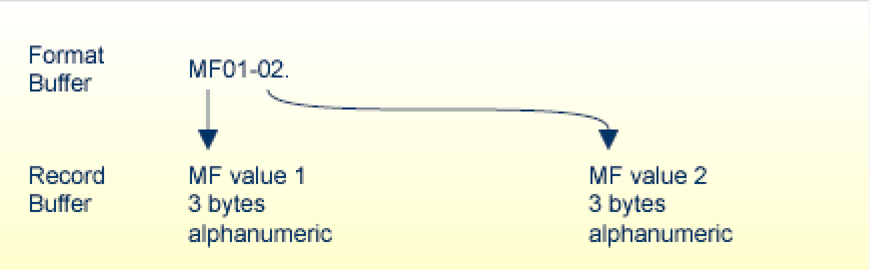
Example 4: The First Two Occurrences of Periodic Group GB



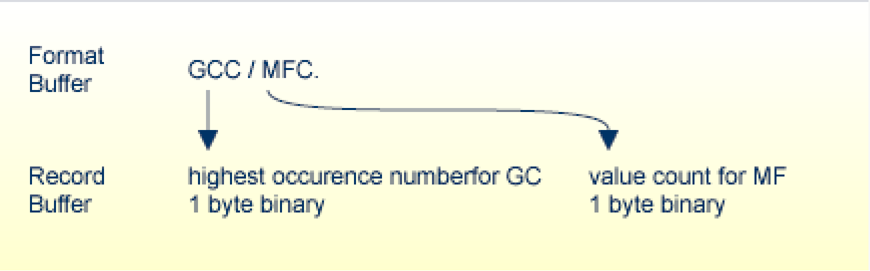
Example 5: The Sixth Value of the Multiple-Value Field MF



Example 6: The First Two Values of the Multiple-Value Field MF



Example 7: The Highest Occurrence Number of a Periodic Group GC and the Existing Number of Values for the Multiple-Value Field MF



13 Prefetch Buffers

When using prefetch buffer segments in an ACB call, the Adabas 8 behavior is unchanged from prior Adabas releases. However, the prefetch option *P* is not supported in an ACBX-based direct call.

14 Multifetch Buffers

Multifetch buffers are needed *only* for some Adabas commands run using the ACBX direct call interface; they are not needed for any ACB interface direct calls.

A multifetch buffer defines an area in storage to which Adabas can return the record descriptor elements (RDEs) of multifetched records. This buffer is only required by Adabas commands for which the multifetch option has been activated (by setting Command Option 1 to "M"). RDEs are each 16 bytes long.

A record descriptor element (RDE) has the structure shown in the following table.

Format	Length	Content
All fields unsigned integer, right aligned	4 bytes	Length of this record in record buffer. Records may have different lengths.
	4 bytes	Adabas response for this record. If a nonzero response is given, no record is stored in the record buffer.
	4 bytes	ISN for this record.
	4 bytes	(L9 only) ISN quantity: value count for this descriptor.

When the multifetch option *M* is set in the Command Option 1 field of an ACBX command, Adabas returns all records being read in the specified record buffer segments, based on the format specifications in the corresponding format buffer segments. For each record buffer segment, the corresponding multifetch buffer segment contains multifetch headers describing the records in the record buffer segment.

For BT or ET commands, a multifetch buffer is *not* needed if Command Option 1 is set to "M". In this case, the ISN buffer is used to store the ISNs that need to be removed from the hold queue.

When a multifetch buffer is required, a corresponding format and record buffer are expected as well. If they are not provided, Adabas will create dummy format and record buffers (with length zero) to pair with the multifetch buffer. For complete information about the relationships between the different types of ABD or buffer specifications, read *Understanding the Different Buffer Types*, in the *Adabas Command Reference Guide*.

Multiple multifetch buffers can be specified for an Adabas direct call. For complete information about multifetch processing, read *Multifetch Operation Processing*, elsewhere in this guide.

15 Search Buffers

The search and value buffers are used together to define:

- the search criteria used to select a set of records using a FIND command (S1, S2, S4); and
- the range of values to be traversed by logical sequential read commands (L3/6, L9).

If a search buffer is provided, a value buffer is also expected. If it is not provided, Adabas will create a dummy value buffer to pair with the search buffer. For complete information about the relationships between the different types of ABD or buffer specifications, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

Only one search and value buffer pair should be specified in a single Adabas direct call.

The user provides the search expressions in the search buffer and the values which correspond to the search expressions in the value buffer.

The syntax used for the search buffer depends on the type of search criteria to be employed:

1. Single file search. The search criteria consists of one or more fields contained in a single file;
2. Multiple file search using physically coupled files. The search criteria consists of fields contained in two or more files which have been physically coupled using the ADAINV utility;
3. Search using the soft coupling feature. This feature provides for a combination of search, read, and internal list matching.

A search criteria may also contain one or more fields which are not defined as descriptors. If nondescriptors are used, Adabas performs a read operation to determine which records are to be returned to the user.

Notes:

1. On files containing a large number of records, performing a search using nondescriptors can result in excessive response times.
2. The behavior of nondescriptor searches in Adabas databases differs between mainframe and open systems in regards to null suppression in the fields. In open systems, nondescriptor searches do not return records with null values in a field if the field is null-suppressed (NU); on mainframe systems, the null-suppression (NU) of fields is ignored during nondescriptor searches. At this time, to resolve this problem, we recommend that you remove the null suppression option (NU) for open systems fields, if the fields must be used for a nondescriptor search.

Search Buffer Syntax

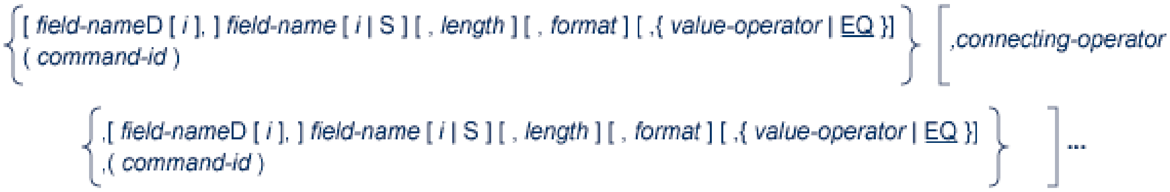
This section outlines the syntax statement options for the search buffer. Delimiters (commas, slashes, parentheses, semicolons) must separate all search buffer entries as indicated. One or more spaces may be present between entries. The search statement must end with a period.

This section covers the following topics:

- Search Expression
- Connecting Search Expressions
- Searching One File
- Searching Multiple, Physically Coupled Files
- Searching One or More Files Using Soft Coupling
- Physically Coupled Files
- Soft Coupling

Search Expression

The search expression syntax is common to all types of searches:



```
{ [ field-nameD [ i ], ] field-name [ i | S ] [ , length ] [ , format ] [ ,{ value-operator | EQ }] }  [ ,connecting-operator
{ ( command-id )                                                                                     }

        { ,[ field-nameD [ i ], ] field-name [ i | S ] [ , length ] [ , format ] [ ,{ value-operator | EQ }] }   ] ...
        { ,( command-id )                                                                                     }
```

Each of the elements in this syntax is now described:

`field-name`

The search expression can name a field (descriptor or nondescriptor), subdescriptor, superdescriptor, hyperdescriptor, collation descriptor, or phonetic descriptor. When using nondescriptors, multiple-value fields are permitted, but sub-/superfields are not.

If a nondescriptor is used, Adabas reads the entire file in order to determine which records satisfy the search criteria. If only descriptors are used, the inverted lists are used and no reading of records is necessary. Search criteria containing nondescriptors and descriptors may be combined.

If a descriptor field is not initialized and logically falls past the end of the physical record, the inverted list entry for that record is not generated for performance reasons and therefore, the record will not be returned in a search. 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 initialize the field for each record of the file.

If the descriptor is defined with the NU option (null-value suppression), null values are not stored in the inverted lists; therefore, a search for all the records which have the null value will always result in no records found (even if there are records in Data Storage which contain a null value for the descriptor). This rule also applies to subdescriptors. A superdescriptor value is not stored if any field from which it is derived is defined with the NU option *and* the value of that field is actually null.

Note: Large object (LB) fields cannot be specified in a search buffer, nor can they be specified in format selection criteria.

`command-id`

The command ID value is enclosed in parentheses and identifies a list of ISNs resulting from a previous S*x* command that specified the save-ISN-list option.

`i` **(Occurrence Index)**

The occurrence index (`i`) identifies a particular occurrence of a descriptor or nondescriptor within a periodic group and is used to limit the search to only the values located in the specified occurrence. If no index is provided, the values in all occurrences are searched.

- The index comprises one to five digits; leading zeros are permitted.
- An index is *not* permitted for a descriptor that is a multiple-value field, or a subdescriptor or superdescriptor derived from a multiple-value field. However, if the multiple value field is within a periodic group, the index is allowed but identifies the occurrence within the PE group and not within the multiple-value field.

D Daylight Savings Time Indicator

For fields with the time zone option TZ, a selection for daylight savings time can also be made using a D indicator element similar to that described in the section *Daylight Savings Indicator (D)*, in the format buffer description elsewhere in this guide.

In search buffers, the following rules apply:

- A daylight savings indicator cannot be specified without a corresponding date-time field. The date-time field must be defined with one of the following date-time edit masks: DATETIME, TIMESTAMP, or NATTIME).
- The daylight savings indicator must precede the corresponding date-time field. For example:

```
AAD,AA,14,U,E(DATETIME),GE.
```

Be sure to read *Daylight Savings Indicator Rules*, in the format buffer description elsewhere in this guide for more information about daylight savings indicator usage in format and search buffers.

S (Significance) and Null Indicators

For fields with the SQL null value compression option NC, a selection for "null" or "not null" can also be made using an "S" null indicator element similar to that described in the section *The SQL Significance Indicator and Field Series Notation*.

Note: The NC option cannot be applied to fields with the NU (null-value suppression), FI (fixed storage), MU (multiple-value), or PE (periodic group) options, or to group fields.

The SQL significance ("S") indicator must be added to the field name ("field-nameS") and the corresponding SQL null indicator must be specified in the value buffer.

The following hexadecimal null indicators are allowed as search argument values:

FFFF	select null values
0000	select non-null values

Any other null indicator value causes an Adabas response code 52 (ADARSP052).

The null indicator (hexadecimal FFFF or 0000) has a standard length of two bytes and fixed-point format; this length and format cannot be overridden.

The "S" indicator can only be used with the equals (=) value-operator; using S with any other value operators causes an Adabas response code 61 (ADARSP061).

Examples:

The S significance operator is part of the search argument for the field AA.

```
AAS.
```

Select records with the FN field value of packed +1 and the AA field value of null (undefined):

Search Buffer	`FN  , 2 , P , D ,  AAS.`	Search argument
Value Buffer	`001F               FFFF`	Field value specification

Note: Insignificant null values are not stored in the index. This can cause a search-for-null operation to be quite costly for an application program's performance.

Select records with the FN field value of packed +1 and the AA field value of non-null:

Search Buffer	`FN   , 2 , P , D ,   AAS.`	Search argument
Value Buffer	`001F                 0000`	Field value specification

`length`

The length of the field/descriptor value as provided in the value buffer. If the length is omitted, the value in the value buffer must comply with the standard length of the field/descriptor, as shown in ***Length and Data Format***.

`format`

The format of the field/descriptor value as provided in the value buffer. If the format is omitted, the value in the value buffer must comply with the standard format of the field/descriptor, as shown in ***Length and Data Format***.

`value-operator`

A value-operator indicates the logical operation to be performed between the preceding descriptor and its corresponding value in the value buffer.

The following operators may be specified:

Operator	Description
EQ (or) =	equals
GE	greater than or equal to
GT (or) >	greater than
LE	less than or equal to
LT (or) <	less than
NE	not equal to

If no value-operator is specified, an equals (EQ) operation is assumed.

Examples:

The following search buffer examples show the use of a value-operator:

Example	Description
AA.	AA equals the value specified in the value buffer (the default)
AA,LT.	AA is less than the value specified in the value buffer
AA,GE.	AA is greater than or equal to the value specified in the value buffer.

The following search buffer using the NE (not equal to) operator selects all records with the FN field not equal to "MIKE":

Search Buffer	`FN,4,A,NE.`	Search argument
Value Buffer	`MIKE`	Field value specification

Replacing the NE operator in this example with EQ (equal to) would select only records with FN field with values of "MIKE".

Connecting Search Expressions

A *connecting operator* may be used to connect search expressions. The permissible connecting operators are as follows:

Operator	Description
D	The results of two search expressions are to be combined using a logical AND operation. For example: `AA,D,AB.`

O	The results of two search expressions are to be combined using a logical OR operation. The OR operator may only be used to connect search expressions which use the same descriptor. Here is a valid and an invalid example:
	Valid:
	`AA,O,AA.`
	Invalid (two different descriptors are used):
	`AA,O,AB.`

R	Fields or command IDs that point to ISN lists derived from different descriptors are to be combined using a logical OR operation. For example: `AA,5,A,R,AB,LT.`

S	A FROM-TO range (inclusive) which involves two search expressions. The same descriptor must be used in both expressions. Here is a valid and an invalid example:
	Valid:
	`AA,S,AA.`
	Invalid (two different descriptors are used):
	`AA,S,AB. ↩`

N	Excludes a single value or a range of values from the immediately preceding FROM-TO range. This operator can only be specified in conjunction with the S operator, and must apply to the same

Operator	Description

	field specified in the FROM-TO range. Phonetic descriptors cannot be specified. Here are some valid and invalid examples:
	- Valid:
	`AA,S,AA,N,AA.`
	Invalid (two different descriptors are used):
	`AA,S,AA,N,AB.`
	- Valid:
	`AA,S,AA,N,AA,S,AA.`
	Invalid (two different descriptors are used):
	`AA,S,AA,N,AA,S,AB.`
	`AA,S,AA,N,AA,N,AB.`

Y	The results of any number of D, O, R, S, and N search operations can be combined using a logical AND operation. For example: `AA,D,AB,Y,AA,O,AA,Y,AA,S,AA,N,AA,S,AA.` The Y connecting operator functions like parentheses: only one level is allowed; that is, nested parentheses are not supported. All search expressions connected with the Y operator must apply to the same file.

If different operators are used within a single search buffer argument, the operators are processed in the following sequence:

1. Evaluate all S operations, as described in this documentation.
2. Evaluate all N operations, as described in this documentation.
3. Evaluate all O operations, as described in this documentation.
4. Evaluate all D operations, if needed.
5. Evaluate all R operations, if needed.
6. Evaluate all Y operations, if needed.

Example:

The following search buffer:

```
AA,S,AA,O,AA,D,AB,R,AC,D,AD.
```

is processed in this sequence:

```
( ( (AA,S,AA),O,AA),D,AB),R,(AC,D,AD)
```

The following search buffer:

```
AA,D,AB,Y,AA,O,AA,Y,AA,S,AA,N,AA,S,AA.
```

is processed in this sequence:

```
(AA,D,AB),Y,(AA,O,AA),Y,((AA,S,AA),N,(AA,S,AA))
```

Searching One File

The following syntax statement is relevant when searching fields in a single file:

```
search-expression [{,connecting-operator,search-expression}...] .
```

For the syntax of the `search-expression`, read ***Search Expression***, elsewhere in this section. For information about the `connecting-operator`, read ***Connecting Search Expressions***, elsewhere in this section.

Searching Multiple, Physically Coupled Files

The following syntax statement is relevant for multiple-file searches in which fields from two or more physically coupled files are to be used:

```
/file-x/search-expression[{,connecting-operator,search-expression}...]
{,D,/file-y search-expression/[{,connecting-operator,search-expression}...]}... .
```

where `file-x` and `file-y` are the file numbers of the physically coupled files. For information about search buffer syntax using physically coupled files, see ***Physically Coupled Files***, elsewhere in this section. For the syntax of the `search-expression`, read ***Search Expression***, elsewhere in this section. For information about the `connecting-operator`, read ***Connecting Search Expressions***, elsewhere in this section.

Searching One or More Files Using Soft Coupling

The following syntax statement is relevant for searching one or more files using soft coupling:

```
(m-file,m-field,s-file,s-field[{;m-file,m-field,s-file,s-field ↩
}...])/s-file-x/search-expression[{,connecting-operator,search-expression}... ]
[{,D,/s-file-y/search-expression[{,connecting-operator,search-expression}...]}...] .
```

where m-file and s-file are...., m-field`s-file-x` and `s-file-y` are the file numbers of the softly coupled files.

For information about search buffer syntax using soft coupling, see ***Soft Coupling***, elsewhere in this section. For the syntax of the `search-expression`, read ***Search Expression***, elsewhere in this section. For information about the `connecting-operator`, read ***Connecting Search Expressions***, elsewhere in this section.

Physically Coupled Files

The syntax of the search buffer for a multiple-file search in which fields from two or more physically coupled files are to be used is:

```
/file-x/search-expression[{,connecting-operator,search-expression}...]
{,D,/file-y search-expression/[{,connecting-operator,search-expression}...]}... .
```

The search criteria of the physically coupled files can be specified in any order. The ISN values actually returned are from the coupled file specified by the Adabas control block's file number field; this file is called the "primary" file.

The elements in this syntax are now described:

`file-x` **and** *`file-y`*

The file numbers of the physically coupled files. All files specified must have been previously coupled using the COUPLE function of the ADAINV utility. A file number can appear only once for a given file. The file number must immediately precede its search criteria (consisting of one or more search expressions and appropriate connecting operators). A maximum of five (5) files may be specified in a single search buffer for physically coupled files.

D

The only **connecting operator** allowed between the search criteria of the physically coupled files is the AND (D) symbol.

`search-expression`

A search expression for the associated physically coupled file. For the syntax of the *`search-expression`*, read ***Search Expression***, elsewhere in this section.

`connecting-operator`

A connecting operator to connect the search expressions of the search criteria for an individual physically coupled file. While the connecting operator between search criteria for the physically coupled files must be "D" (AND), the connecting operators between the search expressions that comprise the search criteria for an individual file can be any of the operators described in ***Connecting Search Expressions***, elsewhere in this section.

Example:

Find the ISNs of all the records in file 1 that contain the three-byte (length override) unpacked decimal (format override) value "+20" in their AB fields, and that are coupled to records in file 2 containing the value "ABCDE" for field RB, which has a standard length of ten bytes and an alphanumeric format.

Search Buffer	`/1/AB,3,U,D,/2/RB.`	Search argument

Value Buffer	character-notation
	`020ABCDEbbbbb`
	hex-notation
	`F0F2F0C1C2C3C4C54040404040`
		Field value specification

Soft Coupling

The syntax of search buffer for a search in which soft coupling is to be used is:

```
(m-file,m-field,s-file,s-field[{;m-file,m-field,s-file,s-field ↩
}...])/s-file-x/search-expression[{,connecting-operator,search-expression}... ]
[{,D,/s-file-y/search-expression[{,connecting-operator,search-expression}...]}...] .
```

The elements in this syntax are now described:

m-file, *m-field*

For *m-file*, specify the number of the main file. This file must also be specified in the file number field of the Adabas control block. The final resulting ISN list will include ISNs contained in the main file only.

For *m-field*, specify the field in the main file that is to be used as the soft-coupling link field. This field must be a descriptor, subdescriptor, superdescriptor, or hyperdescriptor. It may not be a long alphanumeric field or be contained within a periodic group.

The combination of *m-file*, *m-field*, *s-file*, and *s-field* specifications comprise a single soft coupling. A maximum of 42 soft-coupling criteria may be specified. All of the soft coupling must be specified in one set of parentheses.

s-file, *s-field*

For *s-file*, specify the file number of the search file; for *s-field*, specify a field within the search file. For each ISN selected from this search file (according to the search criteria), the field specified as *s-field* will be read. The value of the field will then be used to determine which ISNs in the main file have a matching value.

The field may be a descriptor or nondescriptor; it can be a subdescriptor, superdescriptor, hyperdescriptor, or a long alphanumeric field. It must have the same format as the corresponding *m-field*. The standard length may be different. The field may not be contained within a periodic group.

The combination of *m-file*, *m-field*, *s-file*, and *s-field* specifications comprise a single soft coupling. A maximum of 42 soft-coupling criteria may be specified. All of the soft coupling must be specified in one set of parentheses.

`s-file-x` **and** `s-file-y`

The file number of the coupled files for which you want to specify search criteria. A file number can appear only once for a given file. The file number must immediately precede its search criteria (consisting of one or more search expressions and appropriate connecting operators). A maximum of five (5) files may be specified in a single search buffer.

D

The only **connecting operator** allowed between the search criteria of the coupled files is the AND (D) symbol.

`search-expression`

A search expression for the associated coupled file. For the syntax of the `search-expression`, read ***Search Expression***, elsewhere in this section.

`connecting-operator`

A connecting operator to connect the search expressions of the search criteria for an individual coupled file. While the connecting operator between search criteria for the physically coupled files must be "D" (AND), the connecting operators between the search expressions that comprise the search criteria for an individual file can be any of the operators described in ***Connecting Search Expressions***, elsewhere in this section.

16 Value Buffers

The search and value buffers are used together to define:

- the search criteria to select a set of records using a FIND command (S1, S2, S4); and
- the range of values to be traversed by logical sequential read commands (L3/6, L9).

If a value buffer is provided, a search buffer is also expected. If it is not provided, Adabas will create a dummy search buffer to pair with the value buffer. For complete information about the relationships between the different types of ABD or buffer specifications, read ***Understanding the Different Buffer Types***, elsewhere in this guide.

Only one search and value buffer pair should be specified in a single Adabas direct call.

The user provides the search expressions in the search buffer and the values which correspond to the search expressions in the value buffer.

In the value buffer, the user specifies the values for each descriptor specified in the search buffer.

If the search expression is a command ID, no corresponding entry is made in the value buffer.

Value Buffer Syntax

The values provided must be in the same sequence as the corresponding search expressions specified in the search buffer. All values provided must correspond to the standard length and format of the corresponding descriptor unless the user has explicitly overridden the standard length or format in the search buffer.

No intervening blanks or other characters such as a comma can be inserted between values in the value buffer. A period is not required to end the value buffer entry.

SQL Null Values and Indicators

When searching for fields defined with the NC (SQL null not counted) option, the search buffer field definition must contain a null significance (S) indicator and the corresponding value buffer argument value must display a two-byte binary null value indicator. See the section ***S (Significance) and Null Indicators*** for more information and examples of the null value indicator in the value buffer.

Sign Handling

Binary values are treated as unsigned numbers. Fixed-point, unpacked, and packed values are treated as signed numbers. Valid signs which may be provided are described in thissection:

- Fixed Value Signs
- Unpacked Value Signs
- Packed Value Signs

Fixed Value Signs

For fixed values, the sign is contained in bit 0 (high-order bit):

- 0 = positive
- 1 = negative (two's complement)

Here are two fixed value sign examples showing the hexadecimal notation and the decimal equivalent:

```
00000005 = +5
FFFFFFFB = -5
```

Unpacked Value Signs

For unpacked values, the sign is contained in the four high-order bits of the low-order byte:

- C or A or F or E = positive (CAFE)
- B or D = negative (BD)

Here are two unpacked value sign examples showing the hexadecimal notation and the decimal equivalent:

```
F1F2F3  = +123
F1F2D3  = -123
```

Packed Value Signs

For packed values, the sign is contained in the four low-order bits of the low-order byte:

- A or C or E or F = positive
- B or D = negative

If a search value is being provided for a superdescriptor which is derived from a packed field, an F positive sign or a D negative sign must be provided.

Here are two packed value sign examples showing the hexadecimal notation and the decimal equivalent:

```
X'123F' = +123
X'123C' = +123
X'123D' = -123
```

17 Search and Value Buffer Examples

This section contains examples of search and value buffer construction. For the Adabas file definitions used in all the examples in this section, see ***File Definitions Used in Examples***. The values for the value buffer are shown in character and/or hexadecimal notation.

Example 1: Using a Single Search Expression

A search that uses a single search expression.

Select the ISNs of all the records in file 1 that contain the value "12345" for field AA, which has a standard length of eight bytes and numeric format.

Search Buffer	`AA.`	Search argument

Value Buffer	character-notation
	`00012345`
	hex-notation
	`F0F0F0F1F2F3F4F5`
		Field value specification

The same search may be performed using "AA,5." (length override) in the search buffer and the value "12345" (without trailing blanks) in the value buffer.

Example 2: Using Search Expressions Connected by AND

A search that uses two search expressions connected by the AND operator.

Select the ISNs of all the records in file 1 that contain the value "12345678" for the field AA, which has a standard length of eight bytes and numeric format, and the value "+2" for the field AB, which has a standard length of two bytes and packed decimal format.

Search Buffer	`AA,D,AB.`	Search argument

Value Buffer	hex-notation `F1F2F3F4F5F6F7F8002C`	Field value specification

Select the ISNs of all the records in file 1 that contain the value "12345678" for the field AA, which has a standard length of eight bytes and numeric format, and the value "+2" for the field AB, which has a length of three bytes (override) and unpacked decimal (override) format.

Search Buffer	`AA,D,AB,3,U.`	Search argument

Value Buffer	character-notation
	`12345678002`
	hex-notation
	`F1F2F3F4F5F6F7F8F0F0F2`
		Field value specification

This second search produces the same result as the first search, but shows the use of the length and format override in the search buffer.

Example 3: Using Search Expressions Connected by OR

A search that uses three search expressions connected by the OR operator.

Select the ISNs of all the records in file 2 that contain any of the values "284", "285", or "290" for the field XB. Length and format overrides are used.

Search Buffer	`XB,3,U,O,XB,3,U,O,XB,3,U.`	Search argument
Value Buffer	character-notation `284285290` hex-notation	Field value specification

	`F2F8F4F2F8F5F2F9F0`	

Example 4: Using Search Expressions Connected by FROM-TO

A search that uses two search expressions connected by the FROM-TO operator.

Select the ISNs of all the records in file 2 that contain any value in the range "+20" through "+30" for the field XB.

Search Buffer	`XB,S,XB.`	Search argument
Value Buffer	hex-notation `020C030C`	Field value specification

Example 5: Using Search Expression with BUT-NOT

A search that uses three search expressions connected by the FROM-TO and BUT-NOT operators.

Select the ISNs of all the records in file 2 that contain any of the values in the range "+20" through "+30" but not "+27" for the field XB.

Search Buffer	`XB,S,XB,N,XB.`	Search argument
Value Buffer	hex-notation `020C030C027C`	Field value specification

Select the ISNs of all the records in file 2 that contain any of the values in the range "+1" through "+200" or "+500" through "+600" for the field XB.

Search Buffer	`XB,S,XB,O,XB,S,XB.`	Search argument

Value Buffer	hex-notation `001C200C500C600C`	Field value specification

Example 6: Using a Multiple-Value Descriptor

A search in which a multiple-value field is used.

Select the ISNs of all the records in file 1 that contain the value "ABC" for any value of the multiple-value field MF.

Search Buffer	`MF.`	Search argument

Value Buffer	character-notation
	`ABC ↩`
	hex-notation
	`C1C2C3`
		Field value specification

It is not possible to limit the search to a specific occurrence of a multiple-value field. The following search buffer entry is invalid:

Search Buffer	`MF2.`	Search argument

Example 7: Using a Descriptor Within a Periodic Group

A search in which a descriptor within a periodic group is used.

Select the ISNs of all the records in file 1 that contain the value "4" in any occurrence of the descriptor BA (which is contained in a periodic group).

Search Buffer	`BA.`	Search argument
Value Buffer	hex-notation `04`	Field value specification

Select the ISNs of all records in file 1 that contain the value "4" in the third occurrence of the descriptor BA (which is contained within a periodic group).

Search Buffer	`BA3.`	Search argument
Value Buffer	hex-notation `04`	Field value specification

Example 8: Using a Subdescriptor

A search that uses a subdescriptor. SA is a subdescriptor derived from the first four bytes of the field RA.

Select the ISNs of all the records in file 2 that contain the value "ABCD" for the subdescriptor SA.

Search Buffer	`SA.`	Search argument
Value Buffer	hex-notation `C1C2C3C4`	Field value specification

Example 9: Using a Superdescriptor with Alphanumeric Format

A search that uses a superdescriptor with alphanumeric format. SB is a superdescriptor derived from the first eight bytes of the field RA and the first four bytes of the field RB.

Select the ISNs of all the records in file 2 that contain the value "ABCDEFGH1234" for the superdescriptor SB.

Search Buffer	`SB.`	Search argument
Value Buffer	hex-notation `C1C2C3C4C5C6C7C8F1F2F3F4`	Field value specification

Example 10: Using a Superdescriptor with Binary Format

A search that uses a superdescriptor with binary format. SC is a superdescriptor derived from the fields XB and XC.

Select the ISNs of all the records in file 2 that contain the value "+20" for the field XB and the value "123456" for the field XC.

Search Buffer	`SC.`	Search argument
Value Buffer	hex-notation `020FF1F2F3F4F5F6`	Field value specification

Example 11: Using Previously Created ISN Lists

A search that uses previously created ISN lists (identified by their command IDs).

Select the ISNs present in both ISN lists identified by the command IDs "CID1" and "CID2".

Search Buffer	`(CID1),D,(CID2).`	Search argument
Value Buffer	not used	Field value specification

Select the ISNs of all the records in file 1 for which an ISN is present in the ISN list identified by "CID1" and which contain the value "+123" for the field AB.

Search Buffer	`(CID1),D,AB,3,U.`	Search argument

Value Buffer	character-notation
	`123`
	hex-notation
	`F1F2F3`
		Field value specification

Example 12: Using a Value Operator

A search in which a value operator is used.

Select the ISNs of all the records in file 1 that contain a value greater than "+100" for the field AB.

Search Buffer	`AB,3,U,GT.`	Search argument

Value Buffer	character-notation
	`100`
	hex-notation
	`F1F0F0`
		Field value specification

Example 13: Using Both Value and Connecting Operators

A search in which both value and connecting operators are used.

Select the ISNs of all the records in file 1 that contain a value greater than "+100" for the field AB and a value greater than "A" for the field AA.

Search Buffer	`AB,3,U,GT,D,AA,1,GT.`	Search argument

Value Buffer	character-notation
	`100A`
	hex-notation
	`F1F0F0C1`
		Field value specification

Example 14: Using Physically Coupled Files

A search using search expressions that refer to physically coupled files.

Select the ISNs of all the records in file 1 that contain the value "+20" for the field AB and are coupled to records in file 2 that contain the value "ABCDE" for field RB. Length and format override are used for field AB.

Search Buffer	`/1/AB,3,U,D,/2/RB.`	Search argument

Value Buffer	character-notation
	`020ABCDEbbbbb`
	hex-notation
	`F0F2F0C1C2C3C4C54040404040`
		Field value specification

Example 15: Using Single Soft Coupling Criterion and Single Search Criterion

The file for which ISNs will be returned is determined by the file number field.

File Number	4	
Search Buffer	`(4,AB,1,AC)/1/AB,S,AB.`	Search argument
Value Buffer	`-------------`	Field value specification

1. Search file 1 for AB = value as provided in value buffer.
2. For each resulting ISN in file 1, read field AC and internally match the value with the corresponding value list for file 4. The resulting ISN list from file 4 is provided in the ISN buffer.

Example 16: Using Single Soft Coupling Criterion and Multiple Search Criteria

The file for which ISNs will be returned is determined by the file number field. The order in which the criteria are specified is arbitrary.

File Number	1	
Search Buffer	`(1,AA,2,AB)/1/AC,D,AE,D,/2/AF,S,AF.`	Search argument
Value Buffer	`-------------`	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer)
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 1 for AC = ... and AE = ... (values in value buffer).
4. Match resulting ISN lists from steps 2 and 3. The resulting ISNs are provided in the ISN buffer.

Example 17: Using Multiple Soft Coupling Criteria and Multiple Search Criteria

File Number	1	

Search Buffer	`(1,AA,2,AB; 1,AA,5,BA)`
	`/1/AC,D,AE,D,/2/AF,S,AF,D,/5/BC,S,BC.`
	↩
		Search argument

Value Buffer	`-------------`	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer).
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 5 for BC = ... through ... (values in value buffer).
4. For each resulting ISN in file 5, read field BA and internally match the value with the corresponding value list for file 1.
5. Search file 1 for AC = ... and AE = ... (values in value buffer).
6. Match resulting ISN lists from steps 2, 4 and 5. The resulting ISNs are provided in the ISN buffer.

Example 18: Using Multiple Soft Coupling Criteria and Multiple Search Criteria with Physical Coupling

Search Buffer	`(1,AA,2,AB)`
	`/1/AC,D,AE,D,/2/AF,S,AF,D,/5/BC,S,BC.`
		Search argument

Value Buffer	`-------------`	Field value specification

1. Search file 2 for AF = ... through ... (values in value buffer)
2. For each resulting ISN in file 2, read field AB and internally match the value with the corresponding value list for file 1.
3. Search file 5 for BC = ... through ... (values in value buffer)
4. For each resulting ISN in file 5, use the physical coupling lists created by the ADAINV utility to perform ISN matching. This is done since no soft coupling criteria was provided from file 5 to the main file (file 1). If a physical coupling list does not exist, any ISNs from file 5 will not be considered.
5. Search file 1 for AC = ... and AE = ... (values in value buffer).
6. Match resulting ISN lists from steps 2, 4 and 5. The resulting ISNs are provided in the ISN buffer.

18 Date-Time Edit Mask Processing in Format and Search Buffers

Date-time edit masks can be used to describe the layout of a date/time field in format and search buffers. These edit masks can be used to convert values from the record into a date-time format you need. Unlike the other format buffer **edit masks**, date-time edit masks can also be used to update records.

For example, the following format buffer example produces output for field ND in DATE format:

```
...,ND,8,U,E(DATE).
```

If the field is defined in the FDT with format NATDATE, it will be converted in the record buffer to DATE format.

This chapter describes how Adabas handles date-time edit masks in format and search buffers.

A complete reference of the different date-time edit masks possible is provided in *Date-Time Edit Mask Reference*, in the *Adabas DBA Tasks Manual*. Valid values range from 0001-01-00:00:00:00.000000 through 9999:12-31:23:59:59.999999 are allowed.

- A value of zero (0) is allowed for those date-time edit masks where zero is nota valid value. In these cases, the meaning of the value zero is interpreted as "unknown." If the value is specified for an NC field, the significance indicator is set to "-1", independent of a significance indicator provided in the format or record buffer.

  If you try to convert a date-time edit mask with value zero (when zero represents an unknown date-time value) with another date-time edit mask, the result is always zero. If the target is an NC field, the significance indicator is set to "-1".
- Dates before 1582 (when the Gregorian calendar was introduced) are handled as if the Gregorian calendar was valid also before 1582. Gregorian calendar leap year rules also apply.

Adabas allows you to add a date-time edit mask to any field with formats B, F, P, or U in a loaded file. However, the field may have values stored in the database which are not correct for the specified date-time edit mask. For compatibility reasons, Adabas uses the following processing rules:

- If no date-time edit mask is given in the format or search buffer, no date-time conversions and checks are performed.
- If a date-time edit mask is not compatible with the data-time edit mask in the field definition, an error is issued (response code 41, ADARSP041).
- When specifying a date-time edit mask in the format buffer for a store or update command, the value is checked for a correct date-time value. If the field definition and the edit mask do not match, an error is issued (response code 55, ADARSP055).
- If you specify a date-time edit mask in the format buffer for a read command and the field contains an invalid date-time value or the length of the field is not sufficient to store the value, an error is issued (response code 55, ADARSP055).

Depending on the format and length used for UNIXTIME and XTIMESTAMP fields, you may find that only a subset of the range between) and 9999 years are supported:

- With format B for UNIXTIME or XTIMESTAMP fields, you cannot store date-time values before 1970; this would require negative values that are not supported in binary (B) format.
- Formt F with a length of 4 limits date-time values to January 19, 2038.
- The maximum date for UNIX time is in the 23rd centry with format U and a length of 10.

Fields with date-time edit masks should not be used to store time intervals (ranges) because:

- Zero (0) has a special meaning in date-time values.
- Date-time conversions follow the rules of the underlying format and may return wrong results on interval values.

Date-time edit masks and the daylight saving indicator are not allowed in format selection criteria.

19 ISN Buffers

The ISN buffer defines an area in storage to which Adabas can return the ISNs of the records that satisfy the specified search criteria for a command. In addition, if `Command Option 1` for a command is set to "M" (or "P"), assuming these are valid settings for that command, *and* if the command is issued using the ACB direct call interface, the ISN buffer holds the record descriptor elements (RDEs) of multifetched or prefetched records awaiting processing. If, instead, the command is issued using the ACBX direct call interface, the ISN buffer is not used for this purpose; the multifetch buffer is used instead.

When needed, only one ISN buffer should be specified in a direct call.

The four-byte binary ISNs are usually provided in the ISN buffer in ascending sequence. For the S2 or S9 command, the ISNs are provided according to the user-specified sort sequence. If the ISN buffer is not large enough to contain the entire resulting ISN list, Adabas stores the overflow ISNs on the Adabas work data set, if requested. These overflow ISNs can then be retrieved at a later time.

If the resulting ISNs are to be read using the GET NEXT option of the L1 or L4 commands, the ISN buffer is not needed.

The ISN buffer also supplies an ISN list to be used as input when the ET or BT command specifies a `Command Option 1` of "M" (or "P").

20 User Buffers

User buffers can be used to pass data between the ADALNK user exits 1 and 2 (or A and B in Adabas 7 installations) and Adabas nucleus user exit 11 and 4. Callers using the extended Adabas control block (ACBX) can also pass information between their applications and the exits.

The syntax of this input to user exit 11 and 4 depends upon your requirements. Adabas makes no use of your user buffer data in any way. For complete information about Adabas user exits 11 and 4, read *User Exit 11 (General Processing)* and *User Exit 4 (User-Generated Log Data)*, in the *Adabas User, Hyperdescriptor, Collation Descriptor, and SMF Exits Manual*. For complete information about Adalink user exits 1 and 2 (or A and B), read the appropriate Adabas installation documentation.

ADALNK can be configured to allocate a user buffer by creating an ADALNK globals table and coding a nonzero value for the LGBLSET LUINFO parameter. (For more information about the LGBLSET parameters, refer to your *Adabas Installation* documentation.) The user buffer is normally used to pass information between user exits and ADALNK and the nucleus. The user buffer itself is not available directly to the command issuer when it is allocated by ADALNK.

This chapter refers to fields from several different Adabas data structures. In general, you can identify the data structure of a field by the first two or three characters of the field name, which will indicate the DSECT name. Assembly language DSECTs for these structures can be found in the Adabas source library. There is often additional information about the fields in the DSECT.

Data Structure	Assembly Language DSECT
Adabas buffer description (ABD)	ADADBX
Classic Adabas control block (ACB)	ADACB
Extended Adabas control block (ACBX)	ADACBX
Classic Adabas parameter list	APL
Extended Adabas parameter list	APLX
External command queue	CQX
User block	UB
User exit 11 parameter list	EX11PARM

This chapter covers the following topics:

Differentiating Between the ACB and the ACBX

The user buffer format may differ depending on whether the command originated from an ACB or ACBX call. For this reason, it is important that you (or your applications) can differentiate between ACB and ACBX calls.

In ADALNK exits 1 and 2, the distinction between an ACB or ACBX call is determined using the first entry in the parameter list (APL or APLX) pointed to by user block field UBAUPL. The ACB or ACBX can be examined to see if the byte at offset +2 (ACBCMD or ACBXVERT) contains the character “F” (X’C6’ or ADACBX DSECT symbol ACBXVERE). For more information about the differences between ACB and ACBX calls, read ***Differences between the ACB and the ACBX***, elsewhere in this guide.

Internally, Adabas uses only the ACBX and related Adabas buffer descriptions (ABDs). Commands originating from an ACB call are converted to an ACBX call by ADASVC. When a command has been converted, nucleus exits 4 and 11 are passed the address of a read-only copy of the original ACB. In exit 11, this address is in field EX11PACB of EX11PARM. In exit 4, first locate the CQX using the address in R1 offset 12 (X’C’). Field CQXACB will have the ACB address. In either exit, if the pointer is zero the command originated from an ACBX call.

Using the User Buffer with ADALNK User Exits 1 and 2

ADALNK user exit 1 is invoked before a command is sent to its target and user exit 2 is invoked after the command has completed. These exits have the address of a user block (UB) as a parameter. The user block provides the user buffer address and length.

ADALNK will not set the user block flag UBFLAG.UBFINUB for commands issued from most environments. When this is the case the user buffer is found immediately after the user block at label UBUINFO. The first two bytes of the user buffer contain the length of the user buffer including the two byte length field. The length of the user buffer is also accumulated within UBLUINFO. In ADALNK user exit 1, it is not recommended to do so but you may decrease the effective size of the user buffer by decrementing UBLUINFO appropriately, however you must not reduce UBLUINFO by more than the value specified on the LGBLSET LUINFO parameter. An abend occurs if UBLUINFO is increased.

Flag UBFLAG.UBFINUB is used when a command-issuing environment must service multiple users and give each a unique job name, the most common example being Entire Net-Work. Such environments typically use a copy of ADALNK without user exits.

However, should UBFLAG.UBFINUB be set and the command originated from an ACB call, field UBAUINFO has the address of the 2-byte user block physical length. This is followed by the user buffer itself with its own length prefix, whose value includes the length field. If the command originated from an ACBX call, UBAUINFO may instead have the address of a type "U" Adabas buffer description (ABD). When flag UBFLAG.UBFINUB is set, user exits may not change the value of UBLUINFO and should not change anything else in the user block. The user buffer itself can be accessed and updated without restriction.

Here is an example. Suppose ADALNK is configured to allocated a 48-byte (X'30') user buffer. Assuming flag UBFLAG.UBFINUB is not set, you will see the following in ADALNK exits 1 and 2:

ACB(X)	UBLUINFO	UBINFO
ACB	x'0030'	x'0030...'
ACBX	x'0030'	x'0030...'

Note: In the example above, if Adabas Review is installed then UBLUINFO will be the accumulation of x’0030’ and Adabas Review’s own buffer requirements.

Using the User Buffer with Adabas Nucleus Exits 4 and 11

Adabas user exits 4 (command logging) and 11 (command initiation) may access and update the user buffer. One parameter passed to each of these exits is the address of an external command queue (CQX), a read-only copy of the CQE. Field CQXAUI contains the address of an Adabas buffer description (ABD) that describes the first or only user buffer. It contains zero if there are no user buffers.

The user buffer ABD is arranged in the same manner as all ABDs in the nucleus. Field ABDXID in the ABD will have the value ABDEQUI (X'E4' or C'U') indicating this is a user buffer. Field ABDXLOC will have the value ABDXQIND (X'C9' or C'I'), indicating that the buffer address is found in field ABDXADR. The physical size of the user buffer is provided in field ABDXSIZE; the number of bytes copied from the caller's buffer to the nucleus is provided in the field ABDXSEND. If the user buffer is to be returned from the nucleus to the caller, a nonzero value must be set in field ABDXRECV, representing the number of bytes to copy back to the caller’s buffer. The number of bytes provided in ABDXRECV cannot exceed ABDXSIZE.

For complete information about Adabas buffer descriptions, read ***Adabas Buffer Descriptions (ABDs)***, elsewhere in this guide.

This section describes the format of the user buffer when the commands originates from an ACB call and from an ACBX call.

- User Buffer Format for ACB Calls
- User Buffer Format for ACBX Calls

User Buffer Format for ACB Calls

When ADALNK is invoked with a parameter list for a classic ACB call, the user buffer is formatted as it was in Adabas version 7, so it is compatible with that release. The user buffer will have two 2-byte length prefixes. The first is the buffer length exclusive of the first prefix and the second is inclusive of the second prefix. These values are the same. The ABD field ABDXSIZE will show two bytes more than the ADALNK-generated length to allow for the first length prefix.

If certain add-on products such as Adabas Review are being used, the user buffer may have been increased by ADALNK beyond the user-specified size. The user-specified portion comes first, followed by the add-on product extension.

User Buffer Format for ACBX Calls

When ADALNK is invoked with a parameter list for an ACBX call, the user buffer will have a single 2-byte length prefix. This value includes the length prefix itself.

An ACBX call may provide additional user buffers by constructing an Adabas buffer description (ABD) for each additional user buffer and by including the ABD addresses in the ADALNK parameter list. External command queue (CQX) field CQXAUI will point to the first user buffer ABD, which will describe the first caller-provided user buffer. Other user buffer ABDs follow in a contiguous array with the ADALNK-provided user buffer as the last one of that type. Unless the number of user-allocated user buffers can be predetermined, it may not be possible to step through them starting with CQXAUI. In this case it becomes necessary to scan the entire array of ABDs. Exits 4 and 11 provide, as parameters, the address of the first ABD in the array and the total number of ABDs. Refer to the section *Locating the Correct ABD*, in the user exit documentation for more information about accessing the array of ABDs.

Example

In this example, suppose ADALNK is configured to allocate a 48-byte (x'30') user buffer. In user exits 4 and 11 you will see:

ACB(X)	ABDXSIZE	User Buffer
ACB	x'32'	x'0030 0030..x´2E byte long user data .'
ACBX	x'30'	x'0030… x´2E byte long user data ..'

21 Performance Buffers

Performance buffers are used only if you are using Adabas Review. For complete information, read your Adabas Review documentation.

VI Commands

This chapter provides a detailed description of each Adabas command arranged in alphabetical order by command code.

Command Code	Summary	Description
A1	Update record	Update record(s) (hold option)
BT	Backout transaction	Remove database updates for ET logic users
C1	Write checkpoint	Write command ID, PLOG, RABN RABN checkpoint, buffer flush option
C3	Write SYNX-03 checkpoint	Write SYNX-03 checkpoint for exclusive control update users; option to store user data
C5	Write user data to protection log	Write user data on SIBA/PLOG
CL	Close user session	End ET session and update database
E1	Delete record / refresh file	Delete record (hold option) or refresh file
ET	End transaction	End and save current transaction
HI	Hold record	Prevent record update by other users
L1	Read record	Read record of specified ISN
L2	Read physical sequential record	Read records in physical order
L3	Read logical sequential record	Read records in descriptor value order
L4	Read and hold record	Read record and hold, "wait for held record/issue return code" option
L5	Read physical sequential record and hold	Read records in physical order and hold, "wait/issue return code" option
L6	Read logical sequential record and hold	Read records in descriptor value order with "wait/issue return code" option
L9	Read descriptor values	Read the values of a specified descriptor
LF	Read field definition	Read the characteristics of all fields in a file
N1	Add record with Adabas-assigned ISN	Add new database record with ISN assigned by Adabas

Command Code	Summary	Description
N2	Add record with user-assigned ISN	Add new database record with user-assigned ISN
OP	Open user session	Open user session
RC	Release command ID or global format ID	Release one or more command IDs or a global format ID for the issuing user
RE	Read ET user data	Read ET data for this, another, or all users
RI	Release held record and ISN	Release held record and ISN
S1	Find records	Return count and ISNs of records satisfying the search criterion
S2	Find records in user-specified order	Return count of records and ISNs in user-specified order
S4	Find records and hold	Return count and ISNs of records satisfying the search criterion and put first ISN in list on hold
S5	Find coupled ISNs	Return or save a list of coupled ISNs for the specified file
S8	Process ISN lists	Combine two ISN lists from the same file with an AND, OR, or NOT operation
S9	Sort ISN lists	Sort ISN list in ascending ISN or descriptor-specified sequence

Adabas includes some V* and Y* commands, which you may see mentioned in Adabas shutdown statistics or in Adabas Online System (AOS) screens. These commands are used internally by Adabas and Adabas add-on products and should not be used in direct calls in your applications. Should you use them, errors will result.

22 A1 Command: Update Record

The A1 command updates records with a hold option.

Function and Use

The A1 command is used to change the value of one or more fields in a record. The record containing the field (or fields) to be updated is identified by the file number in which it is contained and its ISN. Specify the fields to be updated in the **format buffer** and provide the updating values for these fields in the **record buffer**. Only the fields specified are modified. All other fields in the record remain unchanged.

All necessary updating to the Associator and Data Storage is performed by Adabas.

A hold option is available to place the record in hold status before it is updated.

If the user is operating in multiuser mode, the A1 command will be executed only if the record to be updated is in hold status for the user.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Note: The A4 command from previous Adabas releases is executed as an A1 command.

ACB Interface Direct Call: A1 Command

This section describes ACB-interface direct calls for the A1 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric / binary	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	U
ISN Lower Limit	17-20	binary	--	A[1]
ISN Quantity	21-24	binary	--	A[1]
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
	29-34	--	--	--
Command Option 1 / 2	35-36	alphanumeric	F	U
	37-44	--	--	--
Additions 2	45-48	alphanumeric / binary		A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Notes

1. These fields are used and not reset by Adabas if coupled files are used.

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

A1

Command ID (ACBCID)

If a series of records is to be updated by using a series of A1 calls, and the same fields are specified in the format buffer for each call (such as when updating a set of records resulting from a find command), this field should be set to a non-blank, non-zero value. If the A1 command is used in conjunction with an L1/L4, L2/L5, or L3/L6 command, and the same fields within each record are read and updated, the same command ID used for the read command should be used for the A1 calls. In both cases, this reduces the time needed to process each successive A1 call.

If only a single record is to be updated with a single A1 call, or the format buffer is modified between A1 calls, this field should be set to blanks.

The leftmost byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

The ISN of the record to be updated.

ISN Quantity/Lower Limit (ACBISQ and ACBISL)

These fields are set to nulls following completion of the A1 operation unless coupled files or partial large object (LOB) fields are used.

If coupled files are used, these fields are used by A1 processing and are not reset.

If a partial LOB field is requested and the Command Option 2 field is set to L, the ACBISL field is used to keep track of the current position in the LOB value when multiple A1 calls are made for the field.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Command Option 1 and Command Option 2 (ACBCOP1 and ACBCOP2)

Option	Description
H	An "H" in either the Command Option 1 or Command Option 2 field places the record in hold status before it is updated. If the record is currently being held by another user, the command is placed in wait status until either the record becomes available or the transaction times out-unless option "R" is also specified.
L	An L in the Command Option 2 field indicates that the position within a large object (LOB) field be tracked in the ACBISL (ISN lower limit) field. This option is useful when multiple A1 commands are being issued in a series for a LOB field. This option can only be used if a LOB field is specified in the format buffer using LOB segment notation with asterisk notation in the `bytenum` specification.
R	If this option is specified, it must be specified in the Command Option 1 field and option "H" must be specified in the Command Option 2 field. Option "R" causes Adabas to return response code 145 (ADARSP145) if the record to be held is not available. The command is *not* placed in wait status.

Additions 2 - Length of Compressed Record - (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- If the A1 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3 - Password - (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

If the accessed file is password protected, Adabas sets this field to blanks during command processing to protect the integrity of the password.

Additions 4 - Cipher Code - (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5 - Format ID, Global Format ID - (ACBADD5)

This field may be used to provide a separate format ID to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high order bit of the first byte of the Additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

ACB Examples

For the Adabas file definitions used in all the examples in this section, see *File Definitions Used in Examples*.

- Example 1
- Example 2

Example 1

ISN 4 of file 1 is to be updated with the following values:

Field AA	1234
Field AB	20

Control Block

Command Code	A1	
Command ID	*bbbb* (blanks)	only 1 record is to be updated
File Number	1	
ISN	4	
Format Buffer Length	10	or larger
Record Buffer Length	10	or larger
Additions 3	*bbbbbbbb* (blanks)	file is not security protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	AA,AB,2,U.
Record Buffer	X'F1F2F3F440404040F2F0'

Example 2

A set of records (previously identified with a FIND command) in file 2 are to be updated with the following values:

Field RA	ABCD
Field XB	80
Field XC	0

Control Block

Command Code	A1	
Command ID	ABCD	a non-blank, non-zero command ID is recommended because the same fields in a series of records are being updated
File Number	2	
ISN	n	each ISN resulting from the previous FIND command will be inserted in this field before each call
Format Buffer Length	9	or larger
Record Buffer Length	16	or larger
Additions 3	password	file 2 is security protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,XC.
Record Buffer	X'C1C2C3C440404040080CF0F0F0F0F0F0'

The A1 call is repeated for each ISN which resulted from the previous find command.

ACBX Interface Direct Call: A1 Command

This section describes ACBX-interface direct calls for the A1 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	binary	---	---
Version Indicator	3-4	binary	F	U
	5-6	binary	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	binary	---	---
Response Code	11-12	binary	---	U
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	U
	33-36	---	---	---
ISN Lower Limit	37-40	binary	---	A[1]
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A[1]
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-68	---	---	---

Field	Position	Format	Before Adabas Call	After Adabas Call
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
	169-193	---	---	---

Notes

1. These fields are used and not reset by Adabas if coupled files are used.

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

A1

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

If a series of records is to be updated by using a series of A1 calls, and the same fields are specified in the format buffer for each call (such as when updating a set of records resulting from a find command), this field should be set to a non-blank, non-zero value. If the A1 command is used in conjunction with an L1/L4, L2/L5, or L3/L6 command, and the same fields within each record are read and updated, the same command ID used for the read command should be used for the A1 calls. In both cases, this reduces the time needed to process each successive A1 call.

If only a single record is to be updated with a single A1 call, or the format buffer is modified between A1 calls, this field should be set to blanks.

The leftmost byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

Use this field to specify the ISN of the record to be updated.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit (ACBXISL)

These fields are set to nulls following completion of the A1 operation unless coupled files or partial large object (LOB) fields are used.

If coupled files are used, these fields are used by A1 processing and are not reset.

If a partial LOB field is requested and the Command Option 2 field is set to L, the ACBXISL field is used to keep track of the current position in the LOB value when multiple A1 calls are made for the field.

ISN Quantity (ACBXISQ)

This field is set to nulls following completion of the A1 operation, unless hard-coupled files are used. If coupled files are used, this field is used during A1 operation and is not reset.

Command Option 1 and Command Option 2 (ACBXCOP1 and ACBXCOP2)

Option	Description
H	An "H" in either the Command Option 1 or Command Option 2 places the record in hold status before it is updated. If the record is currently being held by another user, the command is placed in wait status until either the record becomes available or the transaction times out-unless option "R" is also specified.
L	An L in the Command Option 2 field indicates that the position within a large object (LOB) field be tracked in the ACBXISL (ISN lower limit) field. This option is useful when multiple A1 commands are being issued in a series for a LOB field. This option can only be used if a LOB field is specified in the format buffer using LOB segment notation with asterisk notation in the *bytenum* specification.
R	If specified, this setting must be specified in the Command Option 1 field and option "H" must be specified in the Command Option 2 field. Option "R" causes Adabas to return response code 145 (ADARSP145) if the record to be held is not available. The command is *not* placed in wait status.

Additions 3 - Password - (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

If the accessed file is password protected, Adabas sets this field to blanks during command processing to protect the integrity of the password.

Additions 4 - Cipher Code - (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5 - Format ID, Global Format ID - (ACBXADD5)

This field may be used to provide a separate format ID to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high order bit of the first byte of the Additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers should be specified with an A1 command:

- Format Buffer
- Record Buffer

Format Buffer

The fields to be updated must be specified in this buffer. When performing an A1 command, the format buffer *cannot* contain any of the following:

- **Field selection criteria**;
- An **edit mask element**;
- A reference to a subdescriptor or superdescriptor field;
- The same field specified more than once (except a multiple-value field);
- An "-N" type of specification for an multiple-value field, or for fields in a periodic group (for example, ABN or AB1-N).

Any of the above in the **format buffer** will cause a nucleus response of 44 for an A1 command.

Descriptions of the syntax and examples of format buffer construction are provided in ***Format Buffers***, elsewhere in this guide.

Record Buffer

The values to be used for updating are provided in this buffer according to the length and format as specified in the format buffer. For more information, read ***Record Buffers***, elsewhere in this guide.

Additional Considerations

The following additional considerations are applicable for the A1 command:

1. Subdescriptors, superdescriptors, and phonetic descriptors may not be directly updated. To update any of these descriptors, the fields used to derive them must be updated. All corresponding subdescriptor, superdescriptor, or phonetic descriptor values will then be updated by Adabas.
2. Theoretically, the maximum record length permitted is 32767 bytes before compression. The actual maximum is limited by block size restrictions. It is also smaller depending on the size of the LU parameter specified for the Adabas session; the maximum is (LU - format buffer length - 108). The maximum record length after compression is equal to the smaller of either the Data Storage block size - 4 bytes, or the Work block size - 110 bytes.
3. A descriptor value cannot be larger than 253 bytes.
4. If a field is updated using a length override that exceeds the standard length (not permitted if the field is defined with the Fixed Storage option), all subsequent references to this field should specify the length that was used. If a subsequent reference uses the standard length, value truncation of an alphanumeric fields, a response code 55 (ADARSP055) for a numeric field may occur.
5. A multiple-value field or a field within a periodic group may be specified more than once in the format buffer. A multiple-value field may be specified without an index several times, or with different indices. A periodic field must always have different indices.
6. A multiple-value or periodic group count field specified in the format buffer will be ignored by Adabas. The corresponding value in the record buffer will also be ignored. A literal in the format buffer will be ignored by Adabas, and the corresponding positions in the record buffer will also be ignored.
7. If a multiple-value field is updated and the field count must be changed, Adabas updates the multiple-value field count according to the following rules:
   - For a multiple-value field defined with the null suppression (NU) option, the count field is adjusted to reflect the current number of existing non-null values. Null values are completely suppressed.

Field Definition	01,MF,5,A,MU,NU
MF values before update	XXXXX,YYYYY
Format Buffer	MF4.
Record Buffer	ZZZZZ
Result after update	XXXXX,YYYYY,ZZZZZ MF count = 3
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF2.
Record Buffer	*bbbbb* (blanks)
Result after update	XXXXX,ZZZZZ MF count = 2
MF values before update	XXXXX,ZZZZZ
Format Buffer	MF1-2.
Record Buffer	*bbbbbbbbbb* (blanks)
Result after update	Values suppressed MF count = 0

- For a multiple-value field defined without the NU option, the count is adjusted to reflect the current number of existing values (including null values).

Field Definition	01,MF,5,A,MU
MF values before update	XXXXX,YYYYY
Format Buffer	MF4.
Record Buffer	DDDDD
Result after update	XXXXX,YYYYY,b(blank),DDDDD MF count = 4
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF3.
Record Buffer	*bbbbb* (blanks)
Result after update	XXXXX,YYYYY,b (blank) MF count = 3

A maximum of 191 values is permitted for a multiple-value field.

An exception to the rules is when the format buffer contains a multiple-value field *without an index specification*. In this case, only the new values specified are used and all other values are deleted regardless of the content of the value.

Field Definition	01,MF,5,A,MU,NU
MF values before update	XXXXX,YYYYY
Format Buffer	MF.
Record Buffer	AAAAA
Result after update	AAAAA MF count = 1
MF values before update	XXXXX,YYYYY
Format Buffer	MF1.
Record Buffer	AAAAA
Result after update	AAAAA,YYYYY MF count = 2
MF values before update	XXXXX,YYYYY,ZZZZZ
Format Buffer	MF.
Record Buffer	*bbbbb* (blanks)
Result after update	value suppressed MF count = 0

8. If one or more fields contained in a periodic group are updated, Adabas updates the periodic group count, if necessary, according to the following rule:

   The count is adjusted to reflect the highest occurrence number referenced in the format buffer (provided that this occurrence is higher than the existing highest occurrence number).

Field Definitions	01,GB,PE 02,BA,1,B,DE,NU 02,BB,5,P,NU
GB values before update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2
Format Buffer	GB4.
Record Buffer	X'08000000500F'
Result after update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB (3rd occurrence) BA = 0 BB = 0 GB (4th occurrence) BA = 8 BB = 500 GB count = 4

GB values before update	GB (1st occurrence) BA = 5 BB = 20 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2
Format Buffer	GB1.
Record Buffer	X'00000000000F'
Result after update	GB (1st occurrence) BA = 0 BB = 0 GB (2nd occurrence) BA = 6 BB = 25 GB count = 2

A maximum of 191 occurrences is permitted for a periodic group.

9. If a field defined with variable length (no standard length) is specified in the format buffer, the corresponding value in the record buffer must be preceded by a one-byte binary length value that includes the length byte itself.

Field Definitions	01,AA,3,A 01,AB,A
Format Buffer	AA,AB.
Record Buffer	X'F1F2F306F1F2F3F4F5'

Fields AA and AB are to be updated. The new value for AA is "123". The new value for AB (which is a variable-length field) is "12345".

23 BT Command: Back Out Transaction

The BT command removes database updates for ET logic users.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The BT command is used to remove all database modifications (adds, deletes, updates) performed during the user's current logical transaction. This may be necessary because of a program error or the determination that the entire transaction cannot be completed successfully. BT commands may only be issued by ET logic users.

Adabas issues an implicit ET command as the last step in the processing of a BT command. This causes the current data protection block to be physically written to the Adabas Work and the data protection log, and the release of all records which were held during the transaction.

The Command Option 1 field provides the "P" option to place all records listed in the **ISN buffer** in hold status. The "M" (multifetch) option releases a subset of the records held by the current transaction. The records to be released from hold status are specified in the **ISN buffer**. The first fullword in the buffer specifies the number of 8-byte elements following.

The Command Option 2 field provides the "F" (exclude file) option to exclude a single file from backout processing. The updates performed to the file specified will not be backed out. Any records in the file that are in hold status for the user will, however, be released.

ACB Interface Direct Call: BT Command

This section describes ACB interface direct calls for the BT command. It covers the following topics:

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	binary	--	A
File Number *	9-10	binary	F *	U
Response Code	11-12	binary	--	A
	13-16	--	--	--
ISN Lower Limit	17-20	binary	F	U
	21-32	--	--	--
ISN Buffer Length **	33-34	binary	F **	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
	37-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
ISN **	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if Command Option 2 is specified
**	Required only if Command Option 1 is specified
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

BT

Command ID (ACBCID)

In this field, Adabas returns the transaction sequence number of the transaction that has been backed out. The number is returned in binary format.

File Number (ACBFNR)

If a file is to be excluded from backout processing, the number of the file to be excluded must be specified in this field, and option F must be specified in the Command Option 2 field.

If no file is to be excluded (option F is not specified), any value specified in the file number field is disregarded.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

In this field, Adabas returns the response code for the command. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the BT command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45 - 48 (Additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN Lower Limit (ACBISL)

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes). This length is required only if the hold ISNs or multifetch option is used (see the Command Option 1 field description). If the multifetch feature is specified, this value must be less than 32 KB.

Command Option 1: Hold ISNs Option (ACBCOP1)

Note: If multifetch is set with ADARUN PREFETCH=YES, the "P" option is automatically used for ET/BT commands (the "M" option is automatically used for all other commands). You *cannot* override this option setting by using this field.

By default as part of BT command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	Places all or a portion of these ISNs back into hold status. The ISNs to be returned to hold status must be provided in the **ISN buffer**, and the ISN count must be specified in the ISN lower limit field.
M (command-level multifetch)	Releases only a subset instead of all of the ISNs held by the current transaction. The records to be released from hold status are specified in the **ISN buffer**.The first fullword in the buffer specifies the number of 8-byte elements following.

Command Option 2: Exclude File Option (ACBCOP2)

Option	Description
F (exclude file)	Excludes the single file specified in the file number field from backout processing. The updates performed to the specified file are not backed out but committed. Any records in the file that are in hold status for the user are released.
blanks	All files are to be included in backout processing.

ACB Examples

- Example 1
- Example 2
- Example 3

Example 1

The current user transaction is to be backed out. All files are to be included in the backout process.

Control Block

Command Code	BT	
Command Option 1	blank	no ISNs are held
Command Option 2	b	file exclude option not used

Example 2

The current user transaction is to be backed out. Updates made to file 4 are not to be included in the backout process.

Control Block

Command Code	BT	
File Number	4	file 4 to be excluded from backout
Command Option 2	F	file exclude option used

Example 3

The current user transaction is to be backed out. ISNs 1, 2, and 3 which are contained in file 6 are to be placed into hold status.

Control Block

Command Code	BT	
Command Option 1	P	place ISNs into hold status option
Command Option 2	b	file exclude option not used

Buffer Areas

ISN Buffer	
000600000001	ISN 1
000600000002	ISN 2
000600000003	ISN 2

ACBX Interface Direct Call: BT Command

This section describes ACBX interface direct calls for the BT command. It covers the following topics:

- Control Block and Buffer Overview

- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	binary	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	---	A
Database ID*	17-20	numeric	F*	U
File Number*	21-24	numeric	F*	U
	25-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	45-48	---	---	---
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
Command Option 3	51	alphanumeric	F	U
	52-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
	169-193	---	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
ISN **	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if Command Option 2 is specified
**	Required only if Command Option 1 is specified
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

BT

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

In this field, Adabas returns the transaction sequence number of the transaction that has been backed out. The number is returned in binary format.

Database ID (ACBXDBID)

Use this field to specify the database ID of the file to be excluded from backout processing. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to be excluded from backout processing (option F must be specified in the Command Option 2 field).

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If no file is to be excluded (option F is not specified in Command Option 2), any value specified in the file number field is disregarded.

ISN Lower Limit (ACBXISL)

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

Command Option 1: Hold ISNs Option (ACBXCOP1)

By default as part of BT command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	Places all or a portion of these ISNs back into hold status. The ISNs to be returned to hold status must be provided in the ISN buffer, and the ISN count must be specified in the ISN lower limit field.
M (command-level multifetch)	Releases only a subset instead of all of the ISNs held by the current transaction. The records to be released from hold status are specified in the **ISN buffer**. The first fullword in the buffer specifies the number of 8-byte elements following.

Command Option 2: Exclude File Option (ACBXCOP2)

Option	Description
F (exclude file)	Excludes the single file specified in the file number field from backout processing. The updates performed to the specified file are not backed out but committed. Any records in the file that are in hold status for the user are released.
blanks	All files are to be included in backout processing.

Command Option 3: Shared Hold Status (ACBXCOP3)

Option	Description
H	Keeps the record in shared hold status indefinitely. Records in shared hold status at the time of the BT command are kept in shared hold status beyond the end of the transaction until another ET or BT command is issued (without this H option or the prefetch or multifetch options). Any records in exclusive hold status are also changed to shared hold status beyond the end of the transaction. You cannot use option H if the multifetch or prefetch options are used (or, in Adabas on open systems, if all resources owned by the user are to be released via a command option 1 "T" setting).

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN Buffer

If the Command Option 1 field is set to "P", each ISN whose record is to be returned to hold status must be provided as a six-byte binary entry in which:

- the first two bytes specify the number of the file containing the record; and
- the next four bytes contain the ISN of the record to be held.

If the Command Option 1 field is set to "M", only a subset of the records held by the current transaction are to be released. The first fullword in the **ISN buffer** specifies the number of eight-byte elements, and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status (read ***BT/ET Multifetch Processing***).

24 C1 Command: Write a Checkpoint

The C1 command writes the command ID, PLOG, RABN checkpoint, and buffer flush option.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The C1 command is used to request that a checkpoint be taken.

C1 commands are normally only issued by exclusive control update users (who are not using ET logic) or by users who are operating in single-user mode.

Adabas executes an implied C1 command at the beginning of a user program in which exclusive file control updating has been requested.

The result of the C1 command is a checkpoint entry in the Adabas checkpoint table. This checkpoint entry:

- contains the checkpoint identifier (the value provided by the user in the command ID field), and the current data protection log and block number.
- may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

Specifying the "F" (flush buffers) option in the Command Option 1 or Command Option 2 fields forces Adabas to flush the contents of the Adabas buffer pool to external storage at the end of command processing.

ACB Interface Direct Call: C1 Command

This section describes ACB interface direct calls for the C1 command. It covers the following topics:

- ACB Example

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	A
File Number *	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-34	--	--	--
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
	37-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

C1

Command ID (ACBCID)

A non-blank, non-zero value must be entered in this field. This value identifies the checkpoint taken. It is not necessary that each value provided for each checkpoint be unique.

A value of "0000" or "SYNC" may *not* be specified.

The first byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the C1 command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (Additions 2 field), may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Command Option 1 and Command Option 2: Flush Buffers Option (ACBCOP1 and ACBCOP2)

Option	Description
F (flush buffers)	Specify this option in either the Command Option1 or Command Option2 field to force Adabas to flush the contents of the Adabas buffer pool to external storage at the end of command processing.

ACB Example

A checkpoint entry is to be written. The checkpoint is to be identified by the value "UCP4".

Control Block

Command Code	C1	
Command ID	UCP4	checkpoint identifier

ACBX Interface Direct Call: C1 Command

This section describes ACBX interface direct calls for the C1 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	A
Database ID*	17-20	numeric	F	U
	21-48	---	---	---
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
	169-193	---	---	---

ABDs and Buffers

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

C1

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

A non-blank, non-zero value must be entered in this field. This value identifies the checkpoint taken. It is not necessary that each value provided for each checkpoint be unique.

A value of "0000" or "SYNC" may *not* be specified.

The first byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

Command Option 1 and Command Option 2: Flush Buffers Option (ACBXCOP1 and ACBXCOP2)

Option	Description
F (flush buffers)	Specify this option in either the Command Option 1 or Command Option 2 field to force Adabas to flush the contents of the Adabas buffer pool to external storage at the end of command processing.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

25 C3 Command: Write Checkpoint

The C3 command writes a SYNX-03 checkpoint in the Adabas checkpoint file.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The C3 command may be issued only by exclusive control/update users (who are not using ET logic).

The primary function of the C3 command is to write a SYNX-03 checkpoint in the Adabas checkpoint file. This checkpoint entry:

- contains the current data protection log and block number.
- may be used to restore the database (or certain files) to the status in effect at the time the checkpoint was taken. This may be necessary before a program performing exclusive control updating can be rerun or restarted.

If Command Option 2 is specified, the C3 command also stores user data in the Adabas checkpoint file for restart purposes. The stored data may be subsequently read with an OP or RE command.

ACB Interface Direct Call: C3 Command

This section describes ACB interface direct calls for the C3 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
	5-8	--	--	--
File Number **	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-26	--	--	--

Field	Position	Format	Before Adabas Call	After Adabas Call
Record Buffer Length	27-28	binary	F	U
	29-35	--	--	--
Command Option 2	36	alphanumeric	F	U
	37-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

C3

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully.

Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Record Buffer Length (ACBRBL)

This field is used only if Command Option 2 is specified.

The number of bytes of user data to be stored must be specified in this field.

The maximum length that may be specified is 2000 bytes.

Command Option 2: Store User Data (ACBCOP2)

Option	Description
E	Indicates that user data is to be stored in the Adabas checkpoint file when the session terminates. This option is only available if you provided a non-blank, unique user ID during the OP command for the user session.

ACB Example

A user program issues a C3 command and provides user data to be stored in the Adabas checkpoint file.

Control Block

Command Code	C3	
Record Buffer Length	17	17 bytes of user data to be stored
Command Option 2	E	user data is to be stored

Buffer

Record Buffer	EXU-USER ET-DATA

ACBX Interface Direct Call: C3 Command

This section describes ACBX interface direct calls for the C3 command. It covers the following topics:

- Control Block and Buffer Overview

- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
---	13-16	---	---	---
Database ID**	17-20	numeric	F	U
	21-49	---	---	---
Command Option 2	50	alphanumeric	F	U
	51-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
	169-193	---	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but should be included in Adabas call or one will be automatically generated.
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
C3

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Database ID (ACBXDBID)
Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

Command Option 2: Store User Data (ACBXCOP2)

Option	Description
E	Indicates that user data is to be stored in the Adabas checkpoint file when the session terminates. This option is only available if you provided a non-blank, unique user ID during the OP command for the user session.

Error Subcode (ACBXERRC)
If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to a C3 command:

- Format Buffer
- Record Buffer

Format Buffer

A format buffer is not used by the C3 command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

The **record buffer** contains the user data to be stored in the Adabas checkpoint file. The number of bytes actually stored is determined by the value specified in the record buffer length field.

The user data is retained only if you issued an OP command for the user session in which a non-blank, unique user ID was provided. The data is retained until you issue the next C3 or CL command in which user data is provided. If a non-blank, unique user ID was not provided, the data cannot be retrieved in a subsequent session.

26 C5 Command: Write User Data to Protection Log

The C5 command writes user data on SIBA/PLOG . If you have Event Replicator for Adabas installed, however, you can also use the C5 command to send messages from the originating application to one or more Event Replicator Server destinations. For more information on this Event Replicator for Adabas functionality, read your Event Replicator for Adabas documentation.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The C5 command is used to write user data to the Adabas data protection log. This data may be read and displayed using the ADASEL utility. The data that is written has no effect on Adabas recovery processing. The ADASAV and ADARES utilities ignore all data written to the data protection log as a result of a C5 command.

ACB Interface Direct Call: C5 Command

This section describes ACB interface direct calls for the C5 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
	5-8	--	--	--
File Number **	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-26	--	--	--
Record Buffer Length	27-28	binary	F	U
	29-34	--	--	--
Command Option 1***	35	alphanumeric	F	U
	36	--	--	--
Additions 1***	37-44	alphanumeric	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
	45-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). However, if you are using Event Replicator for Adabas and the Command Option 1 field is set to "R", a file number must be specified to identify the file to which the C5 command applies. For more information, read your Event Replicator for Adabas documentation.
***	Only used if you are using Event Replicator for Adabas. For more information, read your Event Replicator for Adabas documentation.
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

C5

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

However, if you are using Event Replicator for Adabas and the Command Option 1 field is set to "R", a file number must be specified to identify the file to which the C5 command applies. For more information, read your Event Replicator for Adabas documentation.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the C5 command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (Additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Record Buffer Length (ACBRBL)

The number of bytes specified in this field will be written to the Adabas data protection log.

The maximum length which may be specified is 2048 bytes.

Command Option 1 (ACBCOP1)

Used only if you are using the Event Replicator for Adabas. Otherwise, this field must be blank. For more information, read your Event Replicator for Adabas documentation.

Additions 1 (ACBADD1)

Used only if you are using the Event Replicator for Adabas. For more information, read your Event Replicator for Adabas documentation.

ACB Example

The information "ULRR0422 UPDATES FOR JANUARY" is to be written to the Adabas data protection log.

Control Block

Command Code	C5
Record Buffer Length	28

Buffer Areas

Record Buffer	ULRR0422 UPDATES FOR JANUARY

ACBX Interface Direct Call: C5 Command

This section describes ACBX interface direct calls for the C5 command. It covers the following topics:

- Control Block and Buffer Overview

- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
	13-16	---	---	---
Database ID**	17-20	numeric	F	U
File Number***	21-24	numeric	F	U
	25-48	---	---	---
Command Option 1***	49	alphanumeric	F	U
	50-56	---	---	---
Additions 1***	57-64	alphanumeric/ binary	F	U
	65-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but should be included in Adabas call or one will be automatically generated.
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). However, if you are using Event Replicator for Adabas and the Command Option 1 field is set to "R", a file number must be specified to identify the file to which the C5 command applies. For more information, read your Event Replicator for Adabas documentation.
***	Only used if you are using Event Replicator for Adabas. For more information, read your Event Replicator for Adabas documentation.
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

C5

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

If you are using Event Replicator for Adabas and the Command Option 2 field is set to "R", use this field to specify the number of the file to which the C5 command should be directed. For more information, read your Event Replicator for Adabas documentation.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

Command Option 1 (ACBXCOP1)

Used only if you are using the Event Replicator for Adabas. Otherwise, this field must be blank. For more information, read your Event Replicator for Adabas documentation.

Additions 1 (ACBXADD1)

Used only if you are using the Event Replicator for Adabas. For more information, read your Event Replicator for Adabas documentation.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to a C5 command:

Format Buffer

A format buffer is not used by the C5 command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

The information to be written to the data protection log is provided in this buffer.

The information written may be selected subsequently using the ADASEL utility by specifying the beginning characters (1 to 30 characters) as originally contained in the record buffer. It is, therefore, recommended that you provide a unique identification of your data in the beginning positions of the record buffer. This will ensure that the data can be properly identified and selected.

27 CL Command: Close User Session

The CL command ends an ET session and updates the database.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The CL command is used to terminate a user session. Software AG recommends that all user programs issue a CL command upon completion of database processing. User programs operating in single-user mode which are performing database updating must issue a CL command to ensure that all updates have been written to the database.

A CL command:

- issues an implicit ET command (ET-logic users only);
- stores user data in an Adabas system file (optional);
- releases all records currently in hold status for the user as well as all command ID entries (and corresponding ISN lists) assigned to the user;
- transfers the user's ET data from the Adabas Work to an Adabas system file. This is done only if a user ID was provided with the OP command; otherwise, any ET data stored during the session is lost.

A CL command issued by a user operating in single-user mode causes a physical close of the database (Associator, Data Storage, Work, and the data protection log). It is therefore not possible for a single-user mode user to follow a CL command with another command (for example, an OP command).

ACB Interface Direct Call: CL Command

This section describes ACB interface direct calls for the CL command. It covers the following topics:

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	binary	--	A
File Number ***	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	--	A
ISN Quantity	21-24	binary	--	A
	25-26	--	--	--
Record Buffer Length	27-28	binary	F *	U
	29-35	--	--	--
Command Option 2	36	alphanumeric	F	U
	37-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format *	**	--
Record *	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if user data is to be stored
**	Not used but must be included in parameter list of call statement if user data to be stored
***	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

CL

Command ID (ACBCID)

For ET logic users, Adabas returns the transaction sequence number of the user's last successfully executed transaction in this field. The number is provided in binary format.

Because the CL command includes an ET command (see ***Function and Use***), it also increments the transaction sequence number by one (regardless of whether it completes a transaction with or without update commands). The incremented transaction sequence number is then returned.

The CL command returns binary zeros if it ends the session of an ET user without update commands.

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

In this field, Adabas returns the response code for the command. Response Code 0 indicates that the command was executed successfully. If the CL command returns a nonzero response code, the rightmost two bytes of the Adabas control block, Additions 2 field (bytes 47 and 48) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN: Number of I/Os (ACBISN)

Adabas returns the number of I/O operations resulting from this session's Adabas calls in this field.

ISN Lower Limit: Number of Commands (ACBISL)

In this field, Adabas returns the number of commands issued by the user during the user session.

ISN Quantity: CPU Time (ACBISQ)

In this field, Adabas returns an estimate of the amount of processor time used by this user for Adabas command processing.

The time is provided in units of 1.048576 seconds.

Record Buffer Length (ACBRBL)

If user data is to be stored in an Adabas system file, the length of the record buffer must be specified in this field. The length specified determines the number of bytes of user data to be stored.

The maximum length which may be specified is 2000 bytes.

If no user data is to be stored, this field is not used.

Command Option 2: Store User Data (ACBCOP2)

Option	Description
E	Indicates that user data is to be stored in an Adabas system file.

ACB Examples

- Example 1
- Example 2

Example 1

The user program has completed all database activity and issues the CL command. No user data is to be stored.

Control Block

Command Code	CL	
Command Option 2	b	no user data is to be stored

Example 2

The user program issues a CL command and provides user data to be stored in an Adabas system file.

Control Block

Command Code	CL	
Record Buffer Length	17	17 bytes of user data to be stored
Command Option 2	E	user data is to be stored

Buffer

Record Buffer	USER 7 NORMAL END

ACBX Interface Direct Call: CL Command

This section describes ACBX interface direct calls for the CL command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	---	A
Database ID***	17-20	numeric	F	U
	21-28	---	---	---
Number of I/Os	29-32	binary	---	A
	33-36	---	---	---
Number of Commands	37-40	binary	---	A
CPU Time	41-48	binary	---	A
	49	---	---	---
Command Option 2	50	alphanumeric	F	U
	51-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format *	**	--
Record *	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if user data to be stored
**	Not used but should be included in Adabas call or one will be automatically generated.
***	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

CL

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

For ET logic users, Adabas returns the transaction sequence number of the user's last successfully executed transaction in this field. The number is provided in binary format.

Because the CL command includes an ET command (see ***Function and Use***), it also increments the transaction sequence number by one (regardless of whether it completes a transaction with or without update commands). The incremented transaction sequence number is then returned.

The CL command returns binary zeros if it ends the session of an ET user without update commands.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

ISN: Number of I/Os (ACBXISN)

Adabas returns the number of I/O operations resulting from this session's Adabas calls in this field.

ISN Lower Limit: Number of Commands (ACBXISL)

In this field, Adabas returns the number of commands issued by the user during the user session.

ISN Quantity: CPU Time (ACBXISQG)

In this field, Adabas returns an estimate of the amount of processor time used by this user for Adabas command processing.

The time is provided in units of 1/4096 microseconds; therefore, the high-order word of this field contains the estimated processor time in units of 1.048576 seconds.

Command Option 2: Store User Data (ACBXCOP2)

Option	Description
E	Indicates that user data is to be stored in an Adabas system file.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to a CL command:

- Format Buffer
- Record Buffer

Format Buffer

A format buffer is not used by the CL command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

User data output from the command is stored in this buffer. The number of bytes actually stored is determined by the value specified in the record buffer length field. The data is retained only if you have issued an OP command in which a nonblank, unique user ID was provided. If so, the data is retained until you issue the next ET or CL command in which user data is provided. If a nonblank user ID was not provided, the data cannot be retrieved in a subsequent session.

28 E1 Command: Delete Record / Refresh File

The E1 command deletes a record with the hold option, or refreshes a file.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The E1 command deletes a record when the record's ISN is specified, or refreshes a file when an ISN value of zero is specified.

You must specify the file number and ISN of the record to be deleted. Adabas deletes the record from Data Storage. If no ISN is specified and the command ID field contains no command ID (that is, is set to spaces) and the specified file was last loaded with the ADALOD parameter PGMREFRESH=YES, the E1 command refreshes the specified file by first deleting all records of the file and reducing the Associator and Data Storage components of the file to a single extent.

Whether deleting a record or refreshing the entire file, the E1 command also makes the necessary changes to the Associator. You cannot perform both a delete record and file refresh in the same E1 command operation.

If the user is operating in multiuser mode and the record to be deleted is not in hold status for the user, Adabas will place the record in hold status for the user. If the record is in hold status for another user, the E1 command is placed in wait status until the record becomes available. If the R option is specified in the Command Option 1 field and the requested record is not available, response code 145 (ADARSP145) is returned.

Note: The E4 command supported in earlier Adabas releases is executed as an E1 command.

ACB Interface Direct Call: E1 Command

This section describes ACB interface direct calls for the E1 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric / binary	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	U
ISN Lower Limit	17-20	binary	--	A[1]
ISN Quantity	21-24	binary	--	A[1]
	25-34	--	--	--
Command Option 1	35	alphanumeric	F	U
	36-48	--	--	--
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
	65-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Notes

1. These fields are used and not reset by Adabas if coupled files are used.

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)
E1

Command ID (ACBCID)
To refresh a file, set this field to blanks and the ISN field to zero. If you set this field to any other value while specifying an ISN field of zero, the E1 command attempts to remove the record for ISN 0 from the specified file, which causes response code 114 (ADARSP114).

File Number (ACBFNR)
Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the E1 command returns a nonzero response code, the rightmost two bytes of the Adabas control block Additions 2 field, bytes 47 and 48, may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)
The ISN of the record to be deleted. If the command ID field contains blanks and this field contains zero, the file specified by the file number field is refreshed. If the command ID field contains non-blanks and this field contains zero, a response code 114 (ADARSP114) occurs.

ISN Quantity/Lower Limit (ACBISQ and ACBISL)
These fields are set to nulls following completion of the E1 command operation only if hard-coupled files are not used. If coupled files are used, these fields are used for E1 command processing and are not reset.

Command Option 1: Response Code 145 (ADARSP145) if Record not Available (ACBCOP1)
If the user is an ET logic user and the record to be deleted is not in hold status for the user, Adabas places the record in hold status for the user. If the record to be deleted is being held by another user, the action taken by Adabas is controlled by the setting of the Command Option 1 field:

Option	Description
R (return)	Causes Adabas to return response code 145 (ADARSP145) if the record to be deleted is not available. The command is not placed in wait status.

Otherwise, Adabas places the E1 command in wait status until either the record becomes available or the transaction times out.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

ACB Examples

- Example 1
- Example 2

Example 1

ISN 4 in file 2 is to be deleted.

Control Block

Command Code	E1	
File Number	2	record to be deleted is in file 2
ISN	4	record with ISN 4 to be deleted
Additions 3	password	file 2 is security-protected

Example 2

The file specified in the file field (4) is to be refreshed

Control Block

Command Code	E1	
File Number	4	refresh file 4
ISN		set to binary zero
Command ID	*bbbb*	set to blanks

The E1 command refreshes file 4.

ACBX Interface Direct Call: E1 Command

This section describes ACBX interface direct calls for the E1 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28		---	---
ISN	29-32	binary	F	U
	33-36		---	---
ISN Lower Limit	37-40	binary	---	A[1]
	41-44		---	---

Field	Position	Format	Before Adabas Call	After Adabas Call
ISN Quantity	45-48	binary	---	A[1]
Command Option 1	49	alphanumeric	F	U
	50-68	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
	85-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

Notes

1. These fields are used and not reset by Adabas if coupled files are used.

ABDs and Buffers

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
E1

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

To refresh a file, set this field to blanks and the ISN field to zero. If you set this field to any other value while specifying an ISN field of zero, the E1 command attempts to remove the record for ISN 0 from the specified file, which causes response code 114 (ADARSP114).

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISL)

The ISN of the record to be deleted. If the command ID field contains blanks and this field contains zero, the file specified by the file number field is refreshed. If the command ID field contains non-blanks and this field contains zero, a response code 114 (ADARSP114) occurs.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

This field is set to nulls following completion of the E1 command operation, unless hard-coupled files are used. If coupled files are used, this field is used for E1 command processing and is not reset.

ISN Quantity (ACBXISQ)

This field is set to nulls following completion of the E1 command operation, unless hard-coupled files are used. If coupled files are used, this field is used for E1 command processing and is not reset.

Command Option 1: Response Code 145 (ADARSP145) if Record not Available (ACBXCOP1)

If the user is an ET logic user and the record to be deleted is not in hold status for the user, Adabas places the record in hold status for the user. If the record to be deleted is being held by another user, the action taken by Adabas is controlled by the setting of the Command Option 1 field:

Option	Description
R (return)	Causes Adabas to return response code 145 (ADARSP145) if the record to be deleted is not available. The command is not placed in wait status.

Otherwise, Adabas places the E1 command in wait status until either the record becomes available or the transaction times out.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

29 ET Command: End Transaction

The ET command ends and saves the current transaction.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The ET command is used to indicate the end of a logical transaction. Each logical transaction should be secured by issuing an ET command. Subsequent restoring or backing out of any later transaction returns the database status to that set by the last successful ET command.

The ET command:

- writes all current data protection information to the Adabas data protection log and Adabas Work for all update commands executed successfully during the transaction. If the current session ends abnormally, Adabas uses this protection information to reapply the updates for this transaction to the Associator and Data Storage during the next session;
- releases all records held by the user during the current transaction. ISN lists which were saved by the user as a result of find commands are not released (an option is also available by which these ISNs can be returned to hold status);
- optionally stores user data in an Adabas system file. This data may be read subsequently with an OP or RE command as part of a program restart procedure;
- returns a unique sequence number for the transaction in the command ID field of this and any following OP or CL commands issued by the same user. This sequence number may be used to identify the last successfully processed transaction for the user.

The successful execution of an ET command guarantees that all the updates performed during the transaction will be applied to the database, regardless of any subsequent user or Adabas session interruption.

If an ET logic user issues an OP command after a system failure or in the middle of a transaction, Adabas automatically issues a BT command.

ACB Interface Direct Call: ET Command

This section describes ACB interface direct calls for the ET command. It covers the following topics:

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	binary	--	A
File Number****	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-16	--	--	--
ISN Lower Limit	17-20	binary	F	U
ISN Quantity	21-24	binary	--	A
	25-26	--	--	
Record Buffer Length	27-28	binary	F *	U
	29-32	--	--	--
ISN Buffer Length **	33-34	binary	F **	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
	37-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format *	***	--
Record *	F	U
ISN **	F	U

where:

F Supplied by user before Adabas call

A Supplied by Adabas

U Unchanged after Adabas call

* Required only if ET data is to be stored

** Required for hold ISN option; optional for multifetch option

*** Not used but must be included in parameter list of call statement if ET data is to be stored

**** A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

-- Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

ET

Command ID (ACBCID)

Adabas returns either binary zeros or the transaction sequence number in this field.

If the ET command completes a transaction without update commands (that is, A1, E1, N1, N2), Adabas returns binary zeros.

Otherwise, Adabas returns the transaction sequence number in binary format. These numbers are assigned in ascending sequence during a given user session, starting with 1.

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the ET command returns a non-zero response code, the rightmost two bytes of the Adabas control block, bytes 45-48 (Additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN Lower Limit (ACBISL)

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

ISN Quantity (ACBISQ)

In this field, Adabas returns the transaction duration time specified in timer units (each unit equals 1.05 seconds).

Record Buffer Length (ACBRBL)

If user data is to be stored in an Adabas system file, the number of bytes of user data to be stored must be specified in this field.

Adabas will store the number of bytes specified in this field. The maximum number of bytes which may be specified is 2000 bytes.

If no user data is to be stored, this field is not used.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes). This length is required only if the hold ISNs or multifetch option is used (see the Command Option 1 field description).

Command Option 1: Hold ISNs Option (ACBCOP1)

Note: If multifetch is set with ADARUN PREFETCH=YES, the "P" option is automatically used for ET/BT commands (the "M" option is automatically used for all other commands). You *cannot* override this option setting by using this field.

By default as part of ET command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	Places all or a portion of the ISNs back into hold status. The ISNs to be returned to hold status must be provided in the **ISN buffer**, and the ISN count must be specified in the ISN lower limit field.
M (command-level multifetch)	Releases only a subset of the records held by the current transaction. The records to be released from hold status are specified in the **ISN buffer**. The first fullword in the ISN buffer specifies the number of eight-byte elements and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status.

Command Option 2: Store User Data (ACBCOP2)

Option	Description
E	Indicates that user data is to be stored in an Adabas system file.

ACB Examples

- Example 1: ET without User Data
- Example 2: ET with User Data
- Example 3: ET with Hold ISN Option

Example 1: ET without User Data

Control Block

Command Code	ET	
Command Option 2	b	no user data is to be stored

Example 2: ET with User Data

Control Block

Command Code	ET	
Record Buffer Length	25	25 bytes of user data to be stored
Command Option 2	E	user data to be stored

Buffer Areas

Record Buffer	USER DATA FOR TRANSACTION

Example 3: ET with Hold ISN Option

Control Block

Command Code	ET	
ISN Lower Limit	3	
ISN Buffer Length	18	
Command Option 1	P	place ISNs in hold status

Buffer Areas

ISN Buffer	
000600000001	ISN 1
000600000002	ISN 2
000600000003	ISN 3

ACBX Interface Direct Call: ET Command

This section describes ACBX interface direct calls for the ET command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	---	A
Database ID****	17-20	numeric	F	U
	21-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
Command Option 3	51	alphanumeric	F	U
	52-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---

Field	Position	Format	Before Adabas Call	After Adabas Call
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format *	***	--
Record *	F	U
ISN **	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only of ET data to be stored
**	Required for hold ISN option; optional for multifetch option
***	Not used but should be included in Adabas call if ET data is to be stored. If not specified, one will be automatically generated.
****	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

ET

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

Adabas returns either binary zeros or the transaction sequence number in this field.

If the ET command completes a transaction without update commands (that is, A1, E1, N1, N2), Adabas returns binary zeros.

Otherwise, Adabas returns the transaction sequence number in binary format. These numbers are assigned in ascending sequence during a given user session, starting with 1.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

ISN Lower Limit (ACBXISL)

If the hold ISNs option is specified, this field must contain the count of six-byte ISN buffer entries.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

ISN Quantity (ACBXISQ)

In this field, Adabas returns the transaction duration time specified in timer units (each unit equals 1.05 seconds).

Command Option 1: Hold ISNs Option (ACBXCOP1)

By default as part of ET command execution, Adabas releases all ISNs currently held by the user.

Option	Description
P	places all or a portion of the ISNs back into hold status. The ISNs to be returned to hold status must be provided in the **ISN buffer**, and the ISN count must be specified in the ISN lower limit field.
M (command-level multifetch)	Releases only a subset of the records held by the current transaction. The records to be released from hold status are specified in the **ISN buffer**. The first fullword in the ISN buffer specifies the number of eight-byte elements and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status.

Command Option 2: Store User Data (ACBXCOP2)

Option	Description
E	Indicates that user data is to be stored in an Adabas system file.

Command Option 3: Shared Hold Status (ACBXCOP3)

Option	Description
H	Keeps the record in shared hold status indefinitely. Records in shared hold status at the time of the ET command are kept in shared hold status beyond the end of the transaction until another ET or BT command is issued (without this H option or the prefetch or multifetch options). Any records in exclusive hold status are also changed to shared hold status beyond the end of the transaction. You cannot use option H if the multifetch or prefetch options are used (or, in Adabas on open systems, if all resources owned by the user are to be released via a command option 1 "T" setting).

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to an ET command:

- Format Buffer
- Record Buffer
- ISN Buffer

Format Buffer

A format buffer is not used by the ET command, but should be included in the Adabas call if ET data is to be stored. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

The user data to be stored in an Adabas system file is provided in this buffer, if ET data is to be stored.

The data is retained until you issue the next ET or CL command containing ET data. The user data will be retained when the user session terminates only if you issued an OP command in which a unique, non-blank user ID was provided.

ISN Buffer

If the Command Option 1 field is set to "P", each ISN to be returned to hold status must be provided as a six-byte binary entry in which the first two bytes contain the number of the file in which the record is contained, and the next four bytes contain the ISN of the record to be held. Neither the file numbers nor the ISNs are checked for validity.

If the Command Option 1 field is set to "M", only a subset of the records held by the current transaction are to be released. The first fullword in the **ISN buffer** specifies the number of eight-byte elements, and each following eight-byte group is interpreted as one file number/ISN identifier of records to be released from hold status. For more information, read *BT / ET Multifetch Processing*, elsewhere in this guide.

30 HI Command: Hold Record

The HI command prevents record update by other users.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The HI command is used to place a record in hold status. This command is used to hold a record for subsequent updating without allowing other users to update the record until it is released.

Specify the file number and ISN of the record to be held.

If the record to be held is currently being held by another user, the action taken by Adabas is controlled by the setting of the Command Option 1 field in the Adabas control block. If the Command Option 1 field:

- contains an "R", Adabas returns response code 145 (ADARSP145) if the record to be held is not available;
- does not contain an "R", Adabas places the user in wait status until the record becomes available, at which time the user is reactivated automatically.

ACB Interface Direct Call: HI Command

This section describes ACB interface direct calls for the HI command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
	5-8	--	--	--
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
	17-34	--	--	--
Command Option 1	35	alphanumeric	F	U
	36-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)
HI

File Number (ACBFNR)
The number of the file that contains the record to be held.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)
In this field, Adabas returns the response code for the command. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the HI command returns a non-zero response code, the rightmost two bytes of Adabas control block bytes 45-48 (Additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)
The ISN of the record to be placed in hold status.

Command Option 1: Response Code 145 (ADARSP145) if Record Not Available (ACBCOP1)
If the record to be held is currently being held by another user, the action taken by Adabas is controlled by the setting of the Command Option 1 field:

Option	Description
R (return)	Causes Adabas to return response code 145 (ADARSP145) if the record to be held is not available. The command is not placed in wait status.

Otherwise, Adabas places the command in wait status until either the record becomes available, at which time command and user are reactivated automatically, or the transaction times out.

ACB Example

The record identified by ISN 3 in file 2 is to be placed in hold status. Control is not to be returned until the record is available.

Control Block

Command Code	HI	
File Number	2	record to be held is in file 2
ISN	3	record with ISN 3 to be held
Command Option 1	b	response code 145 (ADARSP145) option not used

ACBX Interface Direct Call: HI Command

This section describes ACBX interface direct calls for the HI command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
---	13-16	---	---	---
Database ID	17-20	numeric	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	U
	33-48	---	---	---
Command Option 1	49	alphanumeric	F	U
	50	---	---	---
Command Option 3	51	alphanumeric	F	U
	52-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
HI

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

Use this field to specify the ISN of the record to be placed in hold status.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

Command Option 1: Response Code 145 (ADARSP145) if Record Not Available (ACBXCOP1)

If the record to be held is currently being held by another user, the action taken by Adabas is controlled by the setting of the Command Option 1 field:

Option	Description
R (return)	Causes Adabas to return response code 145 (ADARSP145) if the record to be held is not available. The command is not placed in wait status.

Otherwise, Adabas places the command in wait status until either the record becomes available, at which time command and user are reactivated automatically, or the transaction times out.

Command Option 3: Shared Hold Status (ACBXCOP3)

Option	Description
S	Puts the record in shared hold status until the end of the transaction. The record is placed in shared hold status only if the record is not already in hold status.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

31 L1 and L4 Commands: Read / Read and Hold Record

The L1 and L4 commands read a single record from Data Storage.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

Specify the file number and ISN of the record to be read. In addition, indicate in the **format buffer** which fields are to be read. Adabas returns the requested field values in the **record buffer**.

The L4 command performs the same function as the L1 command, but places the record in hold status. If the record to be held is currently being held by another user, the command is placed in wait status until either the record becomes available or the transaction times out. If the L4 command is issued with the Command Option 1 field set to "R", Adabas returns response code 145 (ADARSP145) if the record to be read and held is unavailable.

The first-unused-ISN (F) option specified in the Command Option 2 field returns the next highest unused ISN for the specified file in the ISN field. The F option cannot be used for expanded files.

The multifetch/prefetch option can improve performance by eliminating the time needed for single-record fetches. Multifetching/prefetching can be enabled by specifying "M", "O" (for multifetching) or "P" (for prefetching) in the Command Option 1 field. Refer to the section ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide, for more information.

The GET NEXT (N) option specified in the Command Option 2 field reads the records identified by the ISNs contained in an ISN list (which was created previously by an S*x* command) without the user having to provide the ISN of the record to be read with each L1 or L4 call. Adabas selects the ISN from the list and reads the record identified by that ISN.

The read by ISN sequence (I) option specified in the Command Option 2 field reads a record identified by the ISN you specify in the ISN field. If the ISN specified is not present in the file, the record of the next higher ISN is read, and that record's ISN is returned in the ISN field.

The compressed option (set by specifying "C." in the format buffer) can be used to request that the record read is to be returned in compressed format as it is stored internally by Adabas.

ACB Interface Direct Call: L1 and L4 Commands

This section describes ACB interface direct calls for the L1 and L4 commands. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	U *
ISN Lower Limit	17-20	binary	F	U
	21-24	--	--	--
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
	29-32	--	--	--
ISN Buffer Length **	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
	37-44	--	--	--
Additions 2	45-48	binary / binary	--	A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	--	A
ISN **	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Except for special options
**	The ISN buffer and length are required only if the multifetch or prefetch option is specified
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

L1 or L4

Command ID (ACBCID)

If a series of records is to be read with a series of L1 or L4 calls, and the same fields are to be specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction of the time required to process each L1 or L4 call.

If the GET NEXT option is to be used, the command ID of the ISN list to be used must be specified in this field. The format buffer may not be changed between successive L1 or L4 calls when the GET NEXT option is used.

If only a single record is to be read, or if the format buffer is to be modified between L1 or L4 calls, this field should be set to blanks.

If the command ID value is X'FFFFFFFF', automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X'00000001', and will increment the value by 1 for each L1 or L4 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID and/or global format ID usage.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates an end-of-list condition (applicable only if the GET NEXT option is used).

ISN (ACBISN)

Specify the ISN of the record to be read.

If you specify the GET NEXT (N) option, Adabas selects ISNs from the ISN list identified by the command ID, reads the record of that ISN, and returns the ISN for the next record in this field; any specified ISN value is ignored.

If you specify the ISN sequence (I) option, you must also specify an ISN value in this field. The L1 or L4 command returns that ISN's data record in the record buffer.

If you specify the "F" option, the L1 or L4 command returns the next higher available ISN recorded in the file control block (FCB) in this field.

When a record is read, Adabas returns that record's ISN in this field, regardless of the option selected. With the option "F", this field returns the next higher unused ISN.

ISN Lower Limit: Multifetch Record Count (ACBISL)

If either "M" or "O" (multifetch option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

If a partial LOB field is requested by this command and the Command Option 2 field is set to L, the ACBISL field is used to keep track of the current position in the LOB value when multiple L1 or L4 calls are made for the field.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The actual format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The actual record buffer area defined in the user program must be at least as large as the length specified.

ISN Buffer Length: With Command-Level Multifetch/Prefetch Option Only (ACBIBL)

The ISN buffer length (in bytes). The ISN buffer defined in the user program must be at least as large as the length specified.

Command Option 1 (ACBCOP1)

Option	Description
F (first unused)	Returns the next highest unused ISN for the specified file in the ISN field. The *next unused ISN* is determined by referring to the file control block (FCB). Do not use the F option when reading Adabas expanded files.
M (multifetching)	Multifetch processing is performed for this command.
O (multifetching with the R option)	Multifetch processing and R option processing (see below) is performed for this command.
P (prefetching)	Prefetch processing is performed for this command.
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L4 command is not available.

Specifying the M, O, or P option indicates that the prefetch or multifetch option is to be used for the command. The selected option is active if either the ISN sequence (I) or GET NEXT (N) option is specified in the Command Option 2 field. The multifetch/prefetch option can improve performance by eliminating the time needed for single-record fetches. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Command Option 2 (ACBCOP2)

Option	Description
F (first unused ISN)	Returns the next higher, unused, ISN for the specified file in the ISN field. The *next unused ISN* is determined by referring to the file control block (FCB). Do not use the F option when reading Adabas expanded files.
I (read by ISN sequence)	Reads the record identified by the ISN specified in the ISN field if the ISN is present in the file. If the ISN is not present in the file, the record with the next higher ISN is read and that record's ISN is returned in the ISN field. If the ISN is not present and no higher ISN is present in the file, no record is read and response code 3 (ADARSP003) is returned.
L (track LOB field position)	An L in the Command Option 2 field indicates that the position within a large object (LOB) field be tracked in the ACBISL (ISN lower limit) field. This option is useful when multiple L1/L4 commands are being issued in a series for a LOB field. This option can only be used if a LOB field is specified in the format buffer using LOB segment notation with asterisk notation in the `bytenum` specification.
N (GET NEXT)	Reads the records identified by the ISNs in an ISN list without the user having to provide the ISN of the record to be read with each L1 or L4 call. Adabas selects the ISN from the list and reads the record identified by that ISN. The ISN list to be used must be identified by the command ID field and must have been created previously by an S*x* command. Response code 3 (ADARSP003) is returned when

Option	Description
	all the ISNs in the list have been selected. Refer to the section *ISN List Processing* for more information.

Additions 2: Length of Compressed and Decompressed Record (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

If the L1 or L4 command returns a nonzero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID (ACBADD5)

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

If the high-order (leftmost) bit of the Additions 5 field is not set to 1, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command, Format, and Global Format IDs* for more information and examples.

ACB Examples

For the Adabas file definitions used in all the examples in this section, see ***File Definitions Used in Examples***, elsewhere in this guide.

- Example 1: Reading a Single Record
- Example 2: Reading a Set of Records
- Example 3: Reading a Set of Records Using the GET NEXT Option
- Example 4: Read with Hold
- Example 5: Read Using the Read ISN Sequence Option
- Example 6: Reading Multiple-Value Fields and Periodic Groups

Example 1: Reading a Single Record

ISN 4 in file 1 is to be read. The values for fields AA and AB are to be returned.

Control Block

Command Code	L1	
Command ID	*bbbb* (blanks)	only 1 record is to be read
File Number	1	
ISN	4	
Format Buffer Length	6	or larger
Record Buffer Length	10	or larger
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	*bbbbbbbb* (blanks)	file not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	AA,AB.

Example 2: Reading a Set of Records

A set of records for which the ISNs have been obtained previously by a find command are to be read from file 2. The values for fields RA and XB are to be returned with the value for field XB to be returned with length 3 and format U.

Control Block

Command Code	L1	
Command ID	ABCD	a nonblank CID is recommended for a series of read operations in which the same fields are being read in each record
File Number	2	
ISN	n	ISNs are taken from the ISN list created by the find command*
Format Buffer Length	10	or larger
Record Buffer Length	11	or larger
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	password	file is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,3,U.

* n indicates an ISN from the ISN list which resulted from the find command. The L1 call is repeated for each ISN in the ISN list.

Example 3: Reading a Set of Records Using the GET NEXT Option

The requirement as stated for example 2 may also be satisfied by using the GET NEXT option.

Control Block

Command Code	L1	
Command ID	ABCD	CID of ISN list to be used
File Number	2	
ISN	0	the entire ISN list is to be selected starting with the first ISN in the list
Format Buffer Length	10	or larger
Record Buffer Length	11	or larger
Command Option 2	N	GET NEXT option to be used
Additions 3	password	file is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,XB,3,U.

The L1 call is repeated for each ISN in the ISN list. No changes to the control block are required between L1 calls. Response code 3 (ADARSP003) will be returned when all the ISNs in the list have been selected.

Example 4: Read with Hold

ISN 5 in file 2 is to be read and held for updating. The values for fields XC and XD are to be returned.

Control Block

Command Code	L4	read with hold
Command ID	*bbbb* (blanks)	only 1 record to be read
File Number	2	
ISN	5	
Format Buffer Length	6	or larger
Record Buffer Length	14	or larger
Command Option 1	b	response code 145 (ADARSP145) option not used
Command Option 2	b	GET NEXT or read ISN sequence options not used
Additions 3	password	file is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	XC,XD.

Example 5: Read Using the Read ISN Sequence Option

File 1 is to be read using the read ISN sequence option. The values for fields AA, AB, and AC are to be returned.

Control Block

Command Code	L1	
Command ID	BCDE	nonblank CID is recommended when a series of records for which the same fields are to be returned is to be read
File Number	1	
ISN	1	if ISN 1 is not present, the record of the next higher ISN in the file is read, and the ISN is returned in this field
Format Buffer Length	6	or larger
Record Buffer Length	30	or larger
Command Option 2	I	read ISN sequence option invoked
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	GA,AC.

Adabas returns the ISN of the record which has been read in the ISN field of the control block. The record with the next higher ISN may be read by adding 1 to the ISN field and repeating the L1 command.

Example 6: Reading Multiple-Value Fields and Periodic Groups

The record identified by ISN 2 in file 1 is to be read. The value for field AA, all values for the multiple-value field MF, and all occurrences of the periodic group GB are to be returned.

Alternative 1: For Six or More Occurrences

1. Issue a L1 call to obtain the count of the number of values which exist for MF, and the highest occurrence count for GB.

   Control Block Field

Command Code	L1	
Command ID	*bbbb* (blanks)	only 1 record is to be read
File Number	1	
ISN	2	
Format Buffer Length	8	or larger
Record Buffer Length	2	or larger
Command Option 2	b	GET NEXT or read ISN sequence option not used

Buffer Areas

Format Buffer	MFC,GBC.

2. Assuming that the result of the above L1 call was the record containing four values for MF and six occurrences for GB, repeat the L1 call using the following format buffer:

   ```
   AA,MF01-04,GB01-06
   ```

   For each call, the length of the format and record buffers must be large enough to hold all entries and values.

   When using this procedure to read a series of records, a valid command ID should be used in step 1, and no command ID (blanks) should be used for step 2 since the content of the format buffer may vary with each step 2 call.

Alternative 2: For Fewer Than Six Occurrences

An alternative solution for example 5 usually provides better performance if the number of values/occurrences is small (less than six) in a large percentage of the records.

1. Issue a L1 call in which the counts for MF and GB are requested, plus the expected number of values and occurrences of MF and GB, plus field AA.

   Assuming that the expected number of values for MF is 2, and the expected number of occurrences of GB is 3, the format buffer for step 1 would be:

   ```
   AA,MFC,GBC,MF01-02,GB01-03.*
   ```

   * Maximum performance is normally achieved if a number which will retrieve all of the values/occurrences in 90 per cent of the records is specified.

2. If either the count received for MF exceeds 2 or the count received for GB exceeds 3, a format buffer similar to that in step 2 of the previous example could be used to obtain the additional values and/or occurrences. Otherwise, no additional call is required.

ACBX Interface Direct Call: L1 and L4 Commands

This section describes ACBX interface direct calls for the L1 and L4 commands. It covers the following topics:

- Control Block and Buffer Overview

- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	U*
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-48	---	---	---
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
Command Option 3 (L4 only)	51	alphanumeric	F	U
	52-64	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	--	A
Multifetch **	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Except for special options
**	The multifetch buffer is required only if the multifetch option is specified.
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

L1 or L4

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates an end-of-list condition (applicable only if the GET NEXT option is used).

Command ID (ACBXCID)

If a series of records is to be read with a series of L1 or L4 calls, and the same fields are to be specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction of the time required to process each L1 or L4 call.

If the GET NEXT option is to be used, the command ID of the ISN list to be used must be specified in this field. The format buffer may not be changed between successive L1 or L4 calls when the GET NEXT option is used.

If only a single record is to be read, or if the format buffer is to be modified between L1 or L4 calls, this field should be set to blanks.

If the command ID value is X'FFFFFFFF', automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X'00000001', and will increment the value by 1 for each L1 or L4 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID and/or global format ID usage.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

Use this field to specify the ISN of the record to be read.

If you specify the GET NEXT (N) option, Adabas selects ISNs from the ISN list identified by the command ID, reads the record of that ISN, and returns the ISN for the next record in this field; any specified ISN value is ignored.

If you specify the ISN sequence (I) option, you must also specify an ISN value in this field. The L1 or L4 command returns that ISN's data record in the record buffer.

If you specify the F option, the L1 or L4 command returns the next higher available ISN recorded in the file control block (FCB) in this field.

When a record is read, Adabas returns that record's ISN in this field, regardless of the option selected. With the option "F", this field returns the next higher unused ISN.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit: Multifetch Record Count (ACBXISL)

If either "M" or "O" (multifetch option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

If a partial LOB field is requested by this command and the Command Option 2 field is set to L, the ACBXISL field is used to keep track of the current position in the LOB value when multiple L1 or L4 calls are made for the field.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet fully used. Set the high-order part of the ACBXISLG field to binary zeros.

Command Option 1 (ACBXCOP1)

Option	Description
F (first unused)	Returns the next highest unused ISN for the specified file in the ISN field. The *next unused ISN* is determined by referring to the file control block (FCB). Do not use the F option when reading Adabas expanded files.
M (multifetching)	Multifetch processing is performed for this command.
O (multifetching with the R option)	Multifetch processing and R option processing (see below) is performed for this command.
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L4 command is not available.

Specifying the "M" or "O" option indicates that the multifetch option is to be used for the command. The selected option is active if either the ISN sequence (I) or GET NEXT (N) option is specified in the Command Option 2 field. The multifetch option can improve performance by eliminating the time needed for single-record fetches. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

Command Option 2 (ACBXCOP2)

Option	Description
F (first unused ISN)	Returns the next higher, unused, ISN for the specified file in the ISN field. The *next unused ISN* is determined by referring to the file control block (FCB). Do not use the F option when reading Adabas expanded files.
I (read by ISN sequence)	Reads the record identified by the ISN specified in the ISN field if the ISN is present in the file. If the ISN is not present in the file, the record with the next higher ISN is read and that record's ISN is returned in the ISN field. If the ISN is not present and no higher ISN is present in the file, no record is read and response code 3 (ADARSP003) is returned.
L (track LOB field position)	An L in the Command Option 2 field indicates that the position within a large object (LOB) field be tracked in the ACBXISL (ISN lower limit) field. This option is useful when multiple L1/L4 commands are being issued in a series for a LOB field. This option can only be used if a LOB field is specified in the format buffer using LOB segment notation with asterisk notation in the `bytenum` specification.
N (GET NEXT)	Reads the records identified by the ISNs in an ISN list without the user having to provide the ISN of the record to be read with each L1 or L4 call. Adabas selects the ISN from the list and reads the record identified by that ISN. The ISN list to be used must be identified by the command ID field and must have been created

Option	Description
	previously by an S*x* command. Response code 3 (ADARSP003) is returned when all the ISNs in the list have been selected. Refer to the section ***ISN List Processing*** for more information.

Command Option 3: Shared Hold Status (ACBXCOP3)

The following command options are available for L4 commands only.

Option	Description
C	Puts the record in shared hold status for the duration of the read operation.
Q	Puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first. This option is available for the L4 command when command option 2 is set to "N".
S	Puts the record in shared hold status until the end of the transaction.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBXADD5)

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

If the high-order (leftmost) bit of the Additions 5 field is not set to 1, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command, Format, and Global Format IDs*** for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers apply to L1 and L4 commands:

- Format Buffer
- Record Buffer
- ISN Buffer
- Multifetch Buffer

Format Buffer

Qualify the fields to be read in this buffer. Descriptions of the syntax and examples of format buffer construction are provided in ***Defining Buffers***, elsewhere in this guide.

A "C." in the first two positions of the format buffer causes the record to be returned in compressed instead of decompressed format.

Record Buffer

Adabas returns the requested field values in this buffer. All values are returned according to the standard format and length of the field unless the user specifies a different length or format in the format buffer.

ISN Buffer

The ISN buffer is used only if the command-level multifetch option is requested and only if the direct call is made using the **ACB interface**. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide. When multifetching is used, the L1 and L4 commands return descriptor elements, each up to 16 bytes long, in the ISN buffer (see the section ***BT/ET Multifetch Processing***).

Multifetch Buffer

The multifetch buffer is used only if the command-level multifetch option is requested and only if the direct call is made using the **ACBX interface**. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide. When multifetching is used, the L1 and L4 commands return descriptor elements, each up to 16 bytes long, in the multifetch buffer (see the section ***BT/ET Multifetch Processing***).

32 L2 and L5 Commands: Read Physical Sequential

The L2 and L5 commands read records in the sequence in which they are physically located in Data Storage.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The L5 command performs the same function as the L2 command, but places each record read in hold status. If the record to be read and held is currently being held by another user, the user will be placed in wait status until the record becomes available. If the L5 command was issued with the Command Option 1 field set to "R", Adabas returns response code 145 (ADARSP145) if the record is not available.

The L2 and L5 commands do not read records in any particular logical order unless the records were loaded initially in a particular logical sequence and no subsequent update changed this order.

The L2 and L5 commands may be used to read an entire file at optimum speed since no access is required to the Associator (as with the L3 command), and all physical blocks are read in consecutive sequence.

Specify the file to be read and the fields within each record for which values are to be returned. Specify the fields in the **format buffer**. Adabas returns the requested field values in the **record buffer**.

The multifetch/prefetch option allows prior accessing of one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. Multifetching or prefetching can be enabled by specifying "M", "O" (for multifetching), or "P" (for prefetching) in the Command Option 1 field. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide.

The compressed option (set by specifying "C." in the format buffer) causes the record read to be returned in compressed format, that is, as stored internally by Adabas.

ACB Interface Direct Call: L2 and L5 Commands

This section describes ACB interface direct calls for the L2 and L5 commands. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN Lower Limit	17-20	binary	F	U
	21-24	--	--	--
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
	29-32	--	--	--
ISN Buffer Length *	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
	36-44	--	--	--
Additions 2	45-48	binary / binary	--	A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F**	U
Record	--	A
ISN *	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	The ISN buffer and length are required only of the multifetch or prefetch option is specified
**	May contain compress option control characters "C."
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

L2 or L5

Command ID (ACBCID)

This field must be set to a non-blank, non-zero value. It is used by Adabas to provide the records in the correct physical order and to avoid the repetitive interpretation of the format buffer. The value provided must not be modified during any given sequential pass of a file.

The first byte of this field may not be set to hexadecimal 'FF'.

If the command ID value is X'FFFFFFFF', automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X'00000001', and will increment the value by 1 for each L2 or L5 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID or global format ID usage.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates an end-of-file (EOF) condition was detected.

ISN (ACBISN)

If this field is set to zero before the initial L2 or L5 call, the sequential pass will begin with the first record contained in the first physical block of the file.

If this field is set to an ISN value before the initial L2 or L5 call, the sequential pass will begin at the first record physically located after the record identified by the ISN specified. The ISN specified must be present in the file.

This field need not be modified by the user after the initial L2 or L5 call. Adabas returns the ISN of the record which has been read in this field.

ISN Lower Limit: Multifetch Record Count (ACBISL)

If either "M" or "O" (multifetching option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified. If either multifetch option "M" or "O" is specified in the Command Option 1 field, this value must be less than 32 kilobytes.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified. If either multifetch option "M" or "O" is specified in the Command Option 1 field, this value must be less than 32 kilobytes.

ISN Buffer Length: Only with Command-Level Multifetch/Prefetch Option (ACBIBL)

The ISN buffer length (in bytes). The ISN buffer area defined in the user program must be as large as (or larger than) the length specified.

Command Option 1 (ACBCOP1)

To improve performance by eliminating the time needed for single-record fetches, set option M, O, or P to enable multifetch or prefetch processing for the command. Refer to the section *Using the Multifetch/Prefetch Feature* for more information.

Option	Description
M	Enable multifetching
O	Use multifetching with the "R" option described below
P	Use prefetching
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L5 command is not available.

Additions 2: Length of Compressed and Decompressed Record (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;

- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

If the L2 or L5 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security or Adabas SAF Security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBADD5)

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high-order (leftmost) bit of the first byte of the Additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command ID, Format ID, Global Format ID*** for more information and examples.

ACB Examples

- Example 1

- Example 2

Example 1

File 2 is to be read in physical sequential order. All the values for all the fields within each record are to be returned.

Control Block

Command Code	L2	
Command ID	EXL2	nonblank CID required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	3	or larger
Record Buffer Length	49	or larger
Command Option 1	b	return option not used
Additions 3	password	file 2 is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RG.

The L2 call is repeated to obtain each successive record. The ISN field need not be modified between calls.

Example 2

File 2 is to be read in physical sequential order. The values for fields RA, XA, and XB (3 bytes unpacked) are to be returned. Each record read is to be placed in hold status for updating purposes.

Control Block

Command Code	L5	
Command ID	EXL5	nonblank CID is required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	13	or larger
Record Buffer Length	21	or larger
Command Option 1	b	response code 145 (ADARSP145) option not used
Additions 3	password	file 2 is security-protected

Additions 4	*bbbbbbbb* (blanks)	file 2 is not ciphered

Buffer Areas

Format Buffer	RA,XA,XB,3,U.

The L5 call is repeated to obtain each successive record. The ISN field need not be modified between L5 calls.

To complete logical transactions and release held records, ET logic users should issue an ET command. Non-ET users should release held records with an A4, E4, or RI command.

ACBX Interface Direct Call: L2 and L5 Commands

This section describes ACBX interface direct calls for the L2 and L5 commands. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-48	---	---	---
Command Option 1	49	alphanumeric	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
	50	---	---	---
Command Option 3 (L5 only)	51	alphanumeric	F	U
	52-64	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F**	U
Record	--	A
Multifetch *	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	A multifetch buffer is required only if the multifetch option is specified.
**	May contain compress option control characters "C."
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
L2 or L5

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates an end-of-file (EOF) condition was detected.

Command ID (ACBXCID)
This field must be set to a non-blank, non-zero value. It is used by Adabas to provide the records in the correct physical order and to avoid the repetitive interpretation of the format buffer. The value provided must not be modified during any given sequential pass of a file.

The first byte of this field may not be set to hexadecimal 'FF'.

If the command ID value is X'FFFFFFFF', automatic command ID generation will be in effect. In this case, the Adabas nucleus will generate values for command ID beginning with X'00000001', and will increment the value by 1 for each L2 or L5 call. When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID or global format ID usage.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)
Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

If this field is set to zero before the initial L2 or L5 call, the sequential pass will begin with the first record contained in the first physical block of the file.

If this field is set to an ISN value before the initial L2 or L5 call, the sequential pass will begin at the first record physically located after the record identified by the ISN specified. The ISN specified must be present in the file.

This field need not be modified by the user after the initial L2 or L5 call. Adabas returns the ISN of the record which has been read in this field.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit: Multifetch Record Count (ACBXISL)

If either "M" or "O" (multifetching option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the **record** and **multifetch** buffer lengths. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

Command Option 1 (ACBXCOP1)

To improve performance by eliminating the time needed for single-record fetches, set option M, O, or P to enable multifetch or prefetch processing for the command. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

Option	Description
M	Enable multifetching
O	Enable multifetching with the "R" option, described below
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L5 command is not available.

Command Option 3: Shared Hold Status (ACBXCOP3)

This following command options are only available for L5 commands:

Option	Description
C	Puts the record in shared hold status for the duration of the read operation.
Q	Puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first.
S	Puts the record in shared hold status until the end of the transaction.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read *Shared Hold Status*, elsewhere in this guide.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security or Adabas SAF Security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID (ACBXADD5)

This field may be used to provide a separate format ID which is to be used to identify the internal format buffer used for this command, or to provide a global format ID.

As long as the high-order (leftmost) bit of the first byte of the Additions 5 field is not set to 1, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command ID, Format ID, Global Format ID* for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers apply to L2 and L5 commands:

- Format Buffer
- Record Buffer
- ISN Buffer
- Multifetch Buffer

Format Buffer

The fields for which values are to be returned are specified in this buffer. Descriptions of the syntax and examples of format buffer construction are provided in ***Defining Buffers***, elsewhere in this guide.

A "C." in the first two positions of the format buffer indicates that the compressed option is to be in effect. This causes the record to be returned in compressed instead of decompressed format.

The format buffer may not be modified after the initial L2 or L5 call has been issued.

Record Buffer

Adabas returns the requested field values in this buffer. All field values are returned according to the standard length and format of each field, unless the user specifies a different length and/or format in the format buffer.

ISN Buffer

The ISN buffer is used only if the command-level multifetch option is used and if the command is issued using an **ACB interface** direct call. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide. When multifetching is used, the L2 or L5 command returns record descriptor elements, each up to 16 bytes long, in the ISN buffer (see the section ***BT/ET Multifetch Processing***).

Multifetch Buffer

The multifetch buffer is used only if the command-level multifetch option is used and if the command is issued using an **ACBX interface** direct call. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide. When multifetching is used, the L2 or L5 command returns record descriptor elements, each up to 16 bytes long, in the multifetch buffer (see the section ***BT/ET Multifetch Processing***).

Additional Considerations

The following additional considerations are applicable for the L2 and L5 commands:

1. The command ID used with an L2 or L5 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. The same command ID may not be used by the user for another read sequential command until it has been released.
2. You are permitted to update or delete records from a file that you are reading with an L2 or L5 command. Adabas maintains information about the last and next record to be provided to you, and is able to provide the correct next record despite any interim record update or deletion you may perform.
3. If another user is updating the file being read with an L2 or L5 command, it is possible that you will not receive one or more records in the file when reading it with the L2 or L5 command. In addition, you may receive the same record more than once during a given sequential pass of the file.

Special L2 Command Call for Response 145

A special L2 command call can be issued if you receive a response code 145 (ADARSP145). When you execute the L2 command with `ISN=0`, `FNR=-4(X'FFFC)`, and an appropriate format buffer, you can obtain information about the user currently holding the ISN when the response code was issued.

Note: You can also obtain this event log information using the ADADBS DEVENTLOG utility function. For more information, read *DEVENTLOG: Display Adabas Event Log*, in the *Adabas Utilities Manual*.

The ACBX control block for this special call should contain the following settings:

Control Block Item	Setting/Description
Command Code	`L2`
Command ID	nonblank command ID required
File number	`-4(X'FFFC')`
ISN	`0`
Format buffer length	`3` or larger
Record buffer length	`8` or larger

The format buffer should be set according to the FDT of file "-4". For example:

```
'1,AA,2,B,FI'     Event type
'1,AB,2,B,FI'     Event sub type
'1,AC,2,B,FI'     DBID
'1,AD,2,B,FI'     NUCID
'1,AE,4,B,FI'     File Number
'1,AF,2,B,FI'     Response Code
'1,AG,2,B,FI'     Subcode
'1,AH,8,B,FI'     ISN
'1,AT,8,B,FI'     Time of Event
'1,AI,8,A,FI'     Job Name of affected user
'1,AJ,28,A,FI,NV' User ID of affected user
'1,AK,8,A,FI'     ET ID of affected user
'1,AL,8,A,FI'     SAF ID of affected user
'1,AQ,4,B,FI'     TID of affected user
'1,AM,8,A,FI'     Job Name of the causer
'1,AN,28,A,FI,NV' User ID of the causer
'1,AO,8,A,FI'     ET ID of the causer
'1,AP,8,A,FI'     SAF ID of the causer
'1,AR,4,B,FI'     TID of causer
```

Should the result of this special L2 call be response code 3 (ADARSP003), no entry for the user could be found for one of the following situations:

- The user did not get a response code 145 (ADARSP145).
- The ADARUN `INFOBUFFERSIZE` parameter is set to "0", so no Adabas event log was allocated for the run.
- The corresponding entry in the Adabas event log was overwritten. When the Adabas event log fills up, it starts overwriting the oldest records in the log.

33 L3 and L6 Commands: Read Logical Sequential

The L3 and L6 commands are used to read a file in logical sequential order, based on the sequence of the values for a given descriptor.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

Specify the:

- file to be read;
- search descriptor whose values are to control the read sequence (phonetic descriptors and descriptors contained within or derived from a field within a periodic group may not be used);
- field or fields within each record for which values are to be returned.

Each L3 or L6 command returns the requested values for one record's field or fields in the **record buffer**. Repeating the command returns the values of each field in order of the search descriptor values.

The ascending and descending options specify whether the records are read in ascending or descending order. The ascending, descending, and value start options may be used to position to a given value before sequential reading begins or to position to a given value during sequential reading. This allows you to position to a specific record in the file without having to read each record.

The multifetch or prefetch options allow prior accessing of one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. Multifetching and prefetching can be enabled by specifying "M" or "O" (for multifetching) in the Command Option 1 field. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

If the format buffer contains "C.", the field values for each record read are returned in compressed format; otherwise, they are returned in uncompressed format.

Processing for the L3 command uses an internal table (the table of sequential commands) to keep track of the most recently returned value and ISN. *Normal* and *expanded* files are operationally diverse and for performance reasons, the value and ISN for each are handled differently in the table. Specifically, the entries for normal files keep the value and ISN that *was returned* on the last call. The entries for expanded files keep the value and ISN that must be returned to the application on the *next* call. This means that for normal files where concurrent updates are taking place, the L3 command returns a newly inserted value/ISN combination as long as it is *greater than or equal to* the last value/ISN returned. For expanded files where concurrent updates are taking place, the L3 command returns a newly inserted value/ISN combination as long as it is *greater than or equal to* the value/ISN that is kept in the table of sequential commands entry as the *next* value/ISN to be returned.

The L6 command performs the same function as the L3 command, but places each record read in hold status. If a record to be read and held is currently being held by another user, the read operation stops and the user is placed in wait status until the record becomes available or an operation timeout occurs. If the Command Option 1 field is set to either "O" or "R" and a requested record is already being held, the L6 command returns response code 145 (ADARSP145).

- Repositioning to a Particular Value
- Changing the Direction of the Read

Repositioning to a Particular Value

For information about specifying a starting value for the sequence-controlling descriptor, read about the ISN control block field in *ISN: Optional Start ISN/Record Read ISN* in either ***ACB Interface Direct Call: L3 and L6 Commands*** or ***ACBX Interface Direct Call: L3 and L6 Commands***. The following procedure is used to *reposition* to a particular value during a sequential pass:

1. Set the last six positions of the Additions 1 field (the descriptor controlling the read sequence) to blanks.
2. Insert in the value buffer the value to which the read sequence is to reposition.
3. Set the ISN field to zero.
4. Set the Command Option 2 field to "A", "D", or "V".

Changing the Direction of the Read

Within a logical read sequence, the direction of the read can be changed at any time from ascending to descending or back without repositioning by specifying "D" or "A" for Command Option 2.

This option is not supported if prefetch or multifetch processing is also requested.

ACB Interface Direct Call: L3 and L6 Commands

This section describes ACB interface direct calls for L3 and L6 commands. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	binary	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN Lower Limit	17-20	binary	F	U
	21-24	--	--	--
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
Search Buffer Length	29-30	binary	F *	U
Value Buffer Length	31-32	binary	F *	U
ISN Buffer Length **	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	A
Additions 2	45-48	binary / binary	--	A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F ***	U
Record	--	A
Search *	F	U
Value *	F	U
ISN **	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if value start option is being used
**	The ISN buffer and length is required only if the multifetch or prefetch option is specified.
***	May contain compress option control characters "C."
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

L3 or L6

Command ID (ACBCID)

This field must be set to a non-blank, non-zero value. The value provided is used by Adabas to provide the records in the correct sequence and to avoid repetitive interpretation of the format buffer.

The first byte of this field may not be set to hexadecimal 'FF', and must not be changed during a given sequential pass of a file.

If the command ID value is X'FFFFFFFF', the command ID is automatically generated. The Adabas nucleus generates command ID values beginning with X'00000001', and increments the value by 1 for each L3 or L6 call.

When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID and/or global format ID usage.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates that an end-of-file condition has been detected.

ISN: Optional Start ISN/Record Read ISN (ACBISN)

For initial or subsequent positioning to a specific location in the logical read sequence when the last six bytes of the Additions 1 field are set to blanks, the ISN field is used together with the starting descriptor value specified in the value buffer to determine the position where reading starts or restarts.

If an existing read sequence is to be continued without repositioning when the last six bytes of the Additions 1 field are not changed from the most recent read command of the sequence, the ISN field is ignored.

Adabas returns in the ISN field the ISN of the record that was read.

When positioning or repositioning in the logical read sequence occurs, the ISN field is not used for positioning if the comparator for the starting descriptor value is set to:

- GT (greater than) for an ascending read sequence; or
- LT (less than) for a descending read sequence.

Otherwise, the ISN field plays a role and the following rules apply:

Rules for Reading Forward (Ascending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying:

- an ISN of zero results in positioning to the *first* (*lowest*) ISN with that descriptor value;
- a non-zero ISN results in positioning to the *lowest* ISN *greater than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *first* ISN of the *next higher* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *first* ISN of the *next higher* descriptor value, regardless of the specified ISN.

Rules for Reading Backward (Descending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying:

- an ISN of zero results in positioning to the *last* (*highest*) ISN with that descriptor value;

- a non-zero ISN results in positioning to the *highest* ISN *less than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *last* ISN of the *next lower* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *last* ISN of the *next lower* descriptor value, regardless of the specified ISN.

ISN Lower Limit: Multifetch Record Count (ACBISL)

If either "M" or "O" (multifetching option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and ISN buffer lengths. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be at least as large as the length specified.

Search Buffer Length (ACBSBL)

The search buffer length (in bytes). The search buffer area defined in the user program must be at least as large as the length specified.

Value Buffer Length (ACBVBL)

The value buffer length (in bytes). The value buffer area defined in the user program must be at least as large as the length specified.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes).

When the Command Option 1 field specifies "P", the L3 and L6 commands use the ISN buffer to hold prefetched descriptor values. The ISN buffer must be large enough to hold the largest descriptor value plus a 16-byte header preceding each value. The actual ISN buffer area defined in the user program must be at least as large as the length specified.

Command Option 1 (ACBCOP1)

Option	Description
M (multifetching)	Multifetch processing is activated.
O (multifetching with the R option)	Multifetch processing as well as option R (see below) processing is activated.
P (prefetching)	Prefetch processing is activated.
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L6 command is not available.

Specifying "M", "O", or "P" indicates that the (command-level) prefetch or multifetch option is to be used. The multifetch/prefetch option can improve performance by allowing prior access to one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. For more information, see the section ***Using the Multifetch/Prefetch Feature***.

Command Option 2 (ACBCOP2)

This option specifies the sequence in which the descriptor's entries are to be processed.

Option	Description
A (ascending)	Processes entries for the descriptor in ascending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.
D (descending)	Processes entries for the descriptor in descending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.
V (value start)	Specifies an *ascending* sequential pass to begin (or continue) at a user-specified value rather than with the lowest (or next) value of the descriptor. The user-specified value *must* be in the value buffer, and the value's length and format *must* be specified in the search buffer. A non-null value specified in the ISN field can further qualify the read operation. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*
blank	Specifies an *ascending* sequential pass without a starting value; all values present are processed. The search buffer and value buffer are ignored when present. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*

Additions 1: Descriptor Used for Sequence Control (ACBADD1)

The descriptor to be used to control the read sequence must be specified in this field. A descriptor, subdescriptor, or superdescriptor may be specified.

- Phonetic descriptors and descriptors contained within or derived from a field contained within a periodic group may not be specified.
- A descriptor which is a multiple-value field may be specified, in which case the same record may be read several times (once for each different value within a given record) during the sequential pass through the file.
- If the descriptor specified is defined with the null value suppression (NU) option, any records containing a null value for the descriptor will not be read.
- The descriptor name is specified in the first two positions of this field. All remaining positions must be set to blanks on the initial call.

The Additions 1 field should not be modified between L3 or L6 calls. The exception is when the user wishes to reposition to a particular value during a sequential pass. This is done by setting the last six positions of this field to blanks, inserting the value to which repositioning is to occur in the value buffer, setting the ISN field to zero, and setting the Command Option 2 field to "A", "D", or "V".

Additions 2: Length of Compressed and Decompressed Record (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

If the L3 or L6 command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are described in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command, Format, and Global Format IDs*** for more information and examples.

ACB Examples

Example 1

File 2 is to be read in logical sequential order. The descriptor RB is to be used for sequence control. The values for the fields RA and RB are to be returned. The entire file is to be read.

Control Block

Command Code	L3	
Command ID	EX01	a non-blank CID is required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Search Buffer Length	0	
Value Buffer Length	0	
Command Option 1	b	response code 145 (ADARSP145) or prefetch options not used
Command Option 2	A	Ascending order with no search or value buffer specified
Additions 1	RB*bbbbbb*	descriptor RB to be used for sequence control, where *bbbbbb* represents blanks.
Additions 3	password	file is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.

The L3 call is repeated to obtain each successive record. The CID and Additions 1 field must not be modified between L3 calls. Response code 3 (ADARSP003) will be returned when all records have been read.

Example 2: Read Logical Sequential with Record Hold

File 1 is to be read in logical sequential order. The descriptor AA is to be used for sequence control. The values for fields AA and AB are to be returned. Each record read is to be placed in hold status.

Control Block

Command Code	L6	
Command ID	EX02	non-blank CID required
File Number	1	
ISN	0	all records to be read
Format Buffer Length	3	or larger
Record Buffer Length	10	or larger
Search Buffer Length	0	
Value Buffer Length	0	
Command Option 1	b	response code 145 (ADARSP145) or prefetch options not used
Command Option 2	b	Ascending order with no search or value buffer specified
Additions 1	AA*bbbbbb*	descriptor AA to be used for sequence control, where *bbbbbb* represents blanks.
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	GA.

To complete logical transactions and release held records, ET logic users should issue an ET command. Non-ET users should release held records with an A4, E4, or RI command.

Example 3: ASCENDING Option

The same requirement as example 1 except that reading is to begin with the value "N".

Control Block

Command Code	L3	
Command ID	EX03	non-blank CID required
File Number	2	
ISN	0	all records are to be read
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger

Search Buffer Length	7	or larger
Value Buffer Length	1	or larger
Command Option 1	b	response code 145 (ADARSP145) option not used
Command Option 2	A	ascending option with start value supplied
Additions 1	RB*bbbbbb*	RB is to be used for sequence control, where *bbbbbb* represents blanks.
Additions 3	password	file is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.	
Search Buffer	RB,1,A.	
Value Buffer	N	reading to begin with value N

The initial L3 call will result in the return of the first record which contains the value "N" for the descriptor RB. If no records exist with this value, the first record which contains a value greater than "N" (such as "NA") will be returned.

Example 4: DESCENDING Option with Repositioning

The same requirement as example 3 except that a value repositioning is to be performed. The sequential read process is to continue with the value "Q".

Control Block

Command Code	L3	
Command ID	EX03	the CID field may not be changed when repositioning
File Number	2	
ISN	0	this field must be set to zeros for value repositioning
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Search Buffer Length	7	or larger
Value Buffer Length	1	or larger
Command Option 1	b	response code 145 (ADARSP145) option not used
Command Option 2	D	descending option with value repositioning
Additions 1	RB*bbbbbb*	the last six positions of this field must be set to blanks for value repositioning
Additions 3	password	file 2 is security-protected
Additions 4	*bbbbbbbb* (blanks)	file 2 is not ciphered

Buffer Areas

Format Buffer	RA,RB.	
Search Buffer	RB,1,A,LE.	
Value Buffer	Q	reposition to value "Q"

ACBX Interface Direct Call: L3 and L6 Commands

This section describes ACBX interface direct calls for L3 and L6 commands. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-48	---	---	---
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
Command Option 3 (L6 only)	51	alphanumeric	F	U
	52-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	A

Field	Position	Format	Before Adabas Call	After Adabas Call
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F ***	U
Record	--	A
Search *	F	U
Value *	F	U
Multifetch **	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Required only if value start option is being used
**	The multifetch buffer is required only if the multifetch option is specified.
***	May contain compress option control characters "C."
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
L3 or L6

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates that an end-of-file condition has been detected.

Command ID (ACBXCID)
This field must be set to a non-blank, non-zero value. The value provided is used by Adabas to provide the records in the correct sequence and to avoid repetitive interpretation of the format buffer.

The first byte of this field may not be set to hexadecimal 'FF', and must not be changed during a given sequential pass of a file.

If the command ID value is X'FFFFFFFF', the command ID is automatically generated. The Adabas nucleus generates command ID values beginning with X'00000001', and increments the value by 1 for each L3 or L6 call.

When specifying user-defined command IDs, the user must ensure that each command ID is unique.

See also the Additions 5 field for separate format ID and/or global format ID usage.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)
Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN: Optional Start ISN/Record Read ISN (ACBXISN)

For initial or subsequent positioning to a specific location in the logical read sequence when the last six bytes of the Additions 1 field are set to blanks, the ISN field is used together with the starting descriptor value specified in the value buffer to determine the position where reading starts or restarts.

If an existing read sequence is to be continued without repositioning when the last six bytes of the Additions 1 field are not changed from the most recent read command of the sequence, the ISN field is ignored.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

Adabas returns in the ISN field the ISN of the record that was read.

When positioning or repositioning in the logical read sequence occurs, the ISN field is not used for positioning if the comparator for the starting descriptor value is set to:

- GT (greater than) for an ascending read sequence; or
- LT (less than) for a descending read sequence.

Otherwise, the ISN field plays a role and the following rules apply:

Rules for Reading Forward (Ascending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying:

- an ISN of zero results in positioning to the *first* (*lowest*) ISN with that descriptor value;
- a non-zero ISN results in positioning to the *lowest* ISN *greater than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *first* ISN of the *next higher* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *first* ISN of the *next higher* descriptor value, regardless of the specified ISN.

Rules for Reading Backward (Descending Option)

If the starting descriptor value specified in the value buffer *is* present in the logical read sequence, specifying:

- an ISN of zero results in positioning to the *last* (*highest*) ISN with that descriptor value;

- a non-zero ISN results in positioning to the *highest* ISN *less than* the specified ISN with that descriptor value, if such an ISN exists. Otherwise, positioning is to the *last* ISN of the *next lower* descriptor value.

If the starting descriptor value specified in the value buffer is *not* present in the logical read sequence, positioning is always to the *last* ISN of the *next lower* descriptor value, regardless of the specified ISN.

ISN Lower Limit: Multifetch Record Count (ACBXISL)

If either "M" or "O" (multifetching option) is specified in the Command Option 1 field, a non-zero value in this field determines the maximum number of records to be multifetched. If this value is zero, the number of records to be multifetched is limited by the record and multifetch buffer lengths. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

Command Option 1 (ACBXCOP1)

Option	Description
M (multifetching)	Multifetch processing is activated.
O (multifetching with the R option)	Multifetch processing as well as option R (see below) processing is activated.
R (return)	Returns response code 145 (ADARSP145) if the record to be read and held by an L6 command is not available.

Specifying the M or O options indicates that the (command-level) multifetch option is to be used. The multifetch option can improve performance by allowing prior access to one or more sequential records, reducing overall operating time and eliminating the time needed for single-record fetches. For more information, see the section ***Using the Multifetch/Prefetch Feature***.

Command Option 2 (ACBXCOP2)

This option specifies the sequence in which the descriptor's entries are to be processed.

Option	Description
A (ascending)	Processes entries for the descriptor in ascending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.
D (descending)	Processes entries for the descriptor in descending sequence with an optional specification of a starting value. A starting descriptor value can be specified in the value buffer. In addition, a non-null value in the ISN field can be used to position within multiple matching descriptor value entries. If the search buffer and value buffer are omitted (SBL and VBL are set to zero (0)), all entries for the descriptor are processed.

Option	Description
V (value start)	Specifies an *ascending* sequential pass to begin (or continue) at a user-specified value rather than with the lowest (or next) value of the descriptor. The user-specified value *must* be in the value buffer, and the value's length and format *must* be specified in the search buffer. A non-null value specified in the ISN field can further qualify the read operation. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*
blank	Specifies an *ascending* sequential pass without a starting value; all values present are processed. The search buffer and value buffer are ignored when present. *This option was used in Adabas prior to version 6; it is maintained to support existing programs.*

Command Option 3: Shared Hold Status (ACBXCOP3)

This following command options are only available for L6 commands:

Option	Description
C	Puts the record in shared hold status for the duration of the read operation.
Q	Puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first.
S	Puts the record in shared hold status until the end of the transaction.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Additions 1: Descriptor Used for Sequence Control (ACBXADD1)

The descriptor to be used to control the read sequence must be specified in this field. A descriptor, subdescriptor, or superdescriptor may be specified.

- Phonetic descriptors and descriptors contained within or derived from a field contained within a periodic group may not be specified.
- A descriptor which is a multiple-value field may be specified, in which case the same record may be read several times (once for each different value within a given record) during the sequential pass through the file.
- If the descriptor specified is defined with the null value suppression (NU) option, any records containing a null value for the descriptor will not be read.
- The descriptor name is specified in the first two positions of this field. All remaining positions must be set to blanks on the initial call.

The Additions 1 field should not be modified between L3 or L6 calls. The exception is when the user wishes to reposition to a particular value during a sequential pass. This is done by setting the last six positions of this field to blanks, inserting the value to which repositioning is to occur in the value buffer, setting the ISN field to zero, and setting the Command Option 2 field to "A", "D", or "V".

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBXADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command, Format, and Global Format ID*** for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers apply to L3 and L6 commands:

- Format Buffer
- Record Buffer
- Search Buffer
- Value Buffer
- Search and Value Buffer Examples
- ISN Buffer
- Multifetch Buffer

Format Buffer

The fields for which values are to be returned must be specified in this buffer. Descriptions of the syntax and examples of format buffer construction are provided in ***Defining Buffers***, elsewhere in this guide.

Specifying "C." in the first two positions indicates that the compressed option is in effect. This causes the record to be returned in compressed instead of decompressed format.

Record Buffer

Adabas returns the requested field values in this buffer. The field values are returned according to the standard format and length of each field, unless the user has requested a different format and/or length in the format buffer.

Search Buffer

Note: Commas separate elements in the search buffer; a period terminates the syntax statement.

The search buffer length (SBL) must be set in the control block. If the search buffer is not used, SBL must be set to zero (0).

If the value start (V) option is used for Command Option 2, the starting value of the descriptor used for sequence control *must* be specified in the value buffer and the value's length and format must be specified in the search buffer.

If the ascending (A) or descending (D) option is used for Command Option 2, a search buffer may be used but is not required. If either the search or value buffer is omitted, all values for a given descriptor are processed. If a starting value, ending value, or both are specified in the value buffer, the search buffer *must* be used to limit the number of descriptor entries retrieved.

The length and format of the descriptor value as provided in the value buffer must be specified in the search buffer if different from the standard length and/or format of the named descriptor.

When a single value is provided in the value buffer (that is, the starting value with option A or the ending value with option D), the syntax for the search buffer is

```
name [ ,length ] [ ,format ] [ ,comparator ]
```

When two values are provided in the value buffer (that is, a starting and an ending value), the syntax for the search buffer is

```
name [ ,length ] [ ,format ] ,S ,name [ ,length ] [ ,format ]
```

The elements used in the search buffer syntax statements are

<table>
<tr><td><b>name</b></td><td>is the name of the descriptor to be used for sequence control. The name specified must be the same as that specified in the Additions 1 field.</td></tr>
<tr><td><b>length</b></td><td>is the length of the value provided in the value buffer. If the length is not specified, it is assumed that the value is being provided using the standard length of the descriptor. See the section <i>Length and Data Format</i> for the allowed length settings.</td></tr>
<tr><td><b>format</b></td><td>is the format of the value provided in the value buffer. If the format is not specified, it is assumed that the value is being provided using the standard format of the descriptor. See the section <i>Length and Data Format</i> for the allowed format settings.</td></tr>
<tr><td><b>comparator</b></td><td>identifies the scope of the read sequence:<table>
<tr><td>GE</td><td>greater than or equal to the value to/from the highest value (the default)</td></tr>
<tr><td>GT</td><td>greater than the value to/from the highest value</td></tr>
<tr><td>LE</td><td>less than or equal to the value to/from the lowest value</td></tr>
<tr><td>LT</td><td>less than the value to/from the lowest value</td></tr>
</table></td></tr>
<tr><td><b>S</b></td><td>A FROM-TO range (inclusive) that involves two search expressions. The same descriptor must be used in both expressions:<table>
<tr><td>AA,S,AA.</td><td>valid</td></tr>
<tr><td>AA,S,AB.</td><td>invalid</td></tr>
</table></td></tr>
</table>

See *Example 3: Overview of Sequence Options* for an overview of sequence options resulting from the choice of A or D for Command Option 2 and various search and value buffer options.

Value Buffer

The value buffer length (VBL) must be set in the control block. If the value buffer is not used, VBL must be set to zero (0).

If the value start option is to be used, or if value repositioning is to be performed, the value with which reading is to start (or continue) must be specified in this buffer.

If the ascending (A) or descending (D) option is used for Command Option 2, a value buffer may be used but is not required. If either the search or value buffer is omitted, all values for a given descriptor are processed. If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

If the ascending (A) option is used for Command Option 2, reading starts (or continues) with the first record containing the value specified. If there are no records with keys equal to a start or reposition (continue) value, reading begins with the first record containing the next higher value.

When two values are provided in the value buffer, the first specifies the lower limit of a range and the second specifies the upper limit. Each value is either the starting or ending value depending on the Command Option 2 setting of A or D.

In addition, if the ISN field is set to a non-zero value, reading will start (or continue) or end with the first record whose ISN is present for the value specified, provided that the ISN is greater than the ISN in the ISN field. If no such ISN exists, the first record with the next higher (or lower) value is read.

Search and Value Buffer Examples

- Example 1: Ascending Option with Optional Search and Value Buffer
- Example 2: Descending Option with Optional Search and Value Buffer
- Example 3: Overview of Sequence Options

Example 1: Ascending Option with Optional Search and Value Buffer

Inverted list for the descriptor used for sequence control:

Value	ISN List
A	1, 4
B	2
D	3, 5

Initial L3 or L6 command with Command Option 2 set to "A" (ascending option), or subsequent L3 or L6 command with Command Option 2 set to "A" and the last six bytes of Additions 1 reset to blanks:

User-Supplied Value	User-Supplied ISN	ISN Where Reading Starts or Continues
A	0	1
A	1	4
A	2	4
A	4	2
A	5	2
B	0	2
B	1	2
B	2	3
B	3	3
BABC	1	3
C	0	3
D	0	3
D	3	5
D	4	5
D	5	response code 3 (ADARSP003)
E-Z	-	response code 3 (ADARSP003)

Example 2: Descending Option with Optional Search and Value Buffer

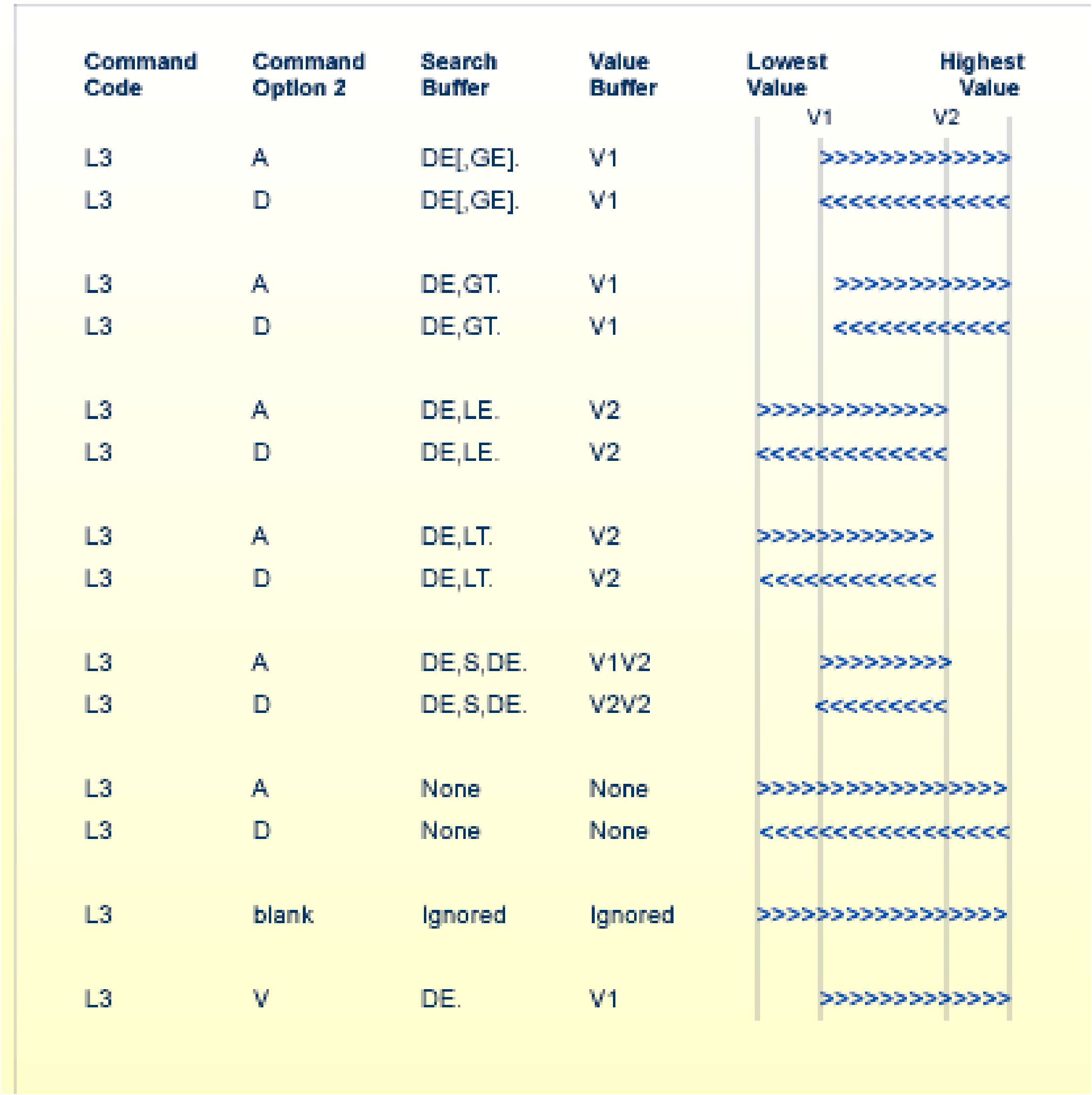
Inverted list for the descriptor used for sequence control:

Value	ISN List
A	1, 9, 25
B	3, 18, 21
C	7, 8, 11

Initial L3 or L6 command with Command Option 2 set to "D" (descending option), ISN=0 for optional start, search buffer comparator set to "LT", and value buffer containing value "C". The start position is the *highest* ISN of the next *lower* descriptor value: in this example, ISN 21 of descriptor value "B".

Example 3: Overview of Sequence Options

The following table illustrates the possibilities for using the ascending/descending option in conjunction with various search buffer and value buffer contents. The following applies to the file being read and ISN=0 (no further positioning within matching start values):



Command Code	Command Option 2	Search Buffer	Value Buffer	Lowest Value V1 V2 Highest Value
L3	A	DE[,GE].	V1	>>>>>>>>>>>>>
L3	D	DE[,GE].	V1	<<<<<<<<<<<<<
L3	A	DE,GT.	V1	>>>>>>>>>>>>
L3	D	DE,GT.	V1	<<<<<<<<<<<<
L3	A	DE,LE.	V2	>>>>>>>>>>>>>
L3	D	DE,LE.	V2	<<<<<<<<<<<<<
L3	A	DE,LT.	V2	>>>>>>>>>>>>
L3	D	DE,LT.	V2	<<<<<<<<<<<<
L3	A	DE,S,DE.	V1V2	>>>>>>>>>
L3	D	DE,S,DE.	V2V2	<<<<<<<<<
L3	A	None	None	>>>>>>>>>>>>>>>>>
L3	D	None	None	<<<<<<<<<<<<<<<<<
L3	blank	Ignored	Ignored	>>>>>>>>>>>>>>>>>
L3	V	DE.	V1	>>>>>>>>>>>>>

Overview of Sequence Options

ISN Buffer

The ISN buffer is used only if the command-level multifetch option is used and if this command is issued using the **ACB direct call interface**. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide.

Multifetch Buffer

The multifetch buffer is used only if the command-level multifetch option is used and if this command is issued using the **ACBX direct call interface**. For more information, read ***Using the Multifetch/Prefetch Feature***, elsewhere in this guide.

Additional Considerations

The following additional considerations are applicable for the L3 or L6 command:

1. The command ID used with the L3 or L6 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. You may not use the same command ID for another read sequential command until the command ID has been released.
2. You are permitted to update or delete records from a file that you are reading with an L3 or L6 command. If the inserted record or records are to be read, you must reposition after having inserted the record or records.
3. If any user is updating the file being read with an L3 or L6 command, it is possible that you may not receive one or more records in the file while reading it with the L3 or L6 command. In addition, you may receive the same record more than once during a sequential pass of the file.

Note: You should avoid updating any descriptor field that controls the read sequence with a value greater than that which it currently contains.

34 L9 Command: Read Descriptor Values

The L9 command reads the values of a specified descriptor.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The L9 command is used to determine the range of values present for a descriptor and the number of records that contain each value.

Specify the file containing the descriptor, the descriptor for which the values are to be returned, and the value at which processing is to begin. Each L9 call returns the next value for the descriptor in the **record buffer**, and the count of records containing that value in the ISN quantity field. The values may be either positive or negative. Null values for descriptors defined with the null value suppression (NU) option are not returned.

The multifetch and prefetch options improve performance by reading several descriptor values at a time. Multifetching can be enabled by specifying "M" (for multifetching) in the Command Option 1 field. Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

The "I" option specified in the Command Option 2 field returns the ISNs of each value in the record buffer. The L9 command reads the Associator inverted lists only; no Data Storage access is required.

Within a logical read sequence, the direction of the read can be changed at any time from ascending to descending or back by specifying "A" or "D" for Command Option 2. See ***Example 3: Overview of Sequence Options*** for an overview of sequence options resulting from the choice of A or D for Command Option 2 and various search and value buffer options. The ascending and descending options are not supported with multifetch processing.

ACB Interface Direct Call: L9 Command

This section describes ACB interface direct calls for the L9 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

- ACB Examples

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	F	A
ISN Quantity	21-24	binary	--	A
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
Search Buffer Length	29-30	binary	F	U
Value Buffer Length	31-32	binary	F	U
ISN Buffer Length *	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
Additions 2	45-48	binary	--	A
Additions 3	49-56	alphanumeric	F	A
	57-64	--	--	--
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	--	A
Search	F	U
Value	F	U
ISN *	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	The ISN buffer and length required only if the multifetch or prefetch option is specified
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

L9

Command ID (ACBCID)

This field must be set to a non-blank, non-zero value. This value is used by Adabas to provide the values in the correct sequence and to avoid the repetitive interpretation of the format buffer.

This field must not be modified during a given sequential pass of the file.

The first byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates that an end-of-file condition has been detected.

ISN: Periodic Group Occurrence (ACBISN)

If the descriptor for which values are to be returned is contained within a periodic group, Adabas returns in this field the occurrence number in which the value being returned is located. The occurrence number is provided in binary format in the two low-order bytes.

If the prefetch option is specified, any occurrence for prefetched values is returned in the header preceding the value in the ISN buffer.

ISN Lower Limit: Lowest ISN in Record Buffer (ACBISL)

Adabas returns values in this field as follows:

Command Option 1	Command Option 2	The L9 command . . .
blank	I	returns the ISNs for each value in the record buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the returned ISN list in the ISN lower limit field.
P (prefetch)	I	returns the first ISN in the record buffer and all prefetched descriptor values in the ISN buffer, preceded by a 16-byte header; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value prefetched in the ISN lower limit field.
M (multifetch)	I	returns the group of multifetched records in the record buffer and a description of these records in the caller's ISN buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value multifetched in the ISN lower limit field.

If Command Option 1 is set to "M" (multifetch option), you can set:

- a non-zero value in the ISN lower limit field to limit the number of values to be multifetched.
- zero in the ISN lower limit field to multifetch all values.

Refer to the section *Using the Multifetch/Prefetch Feature*, elsewhere in this guide, for more information.

ISN Quantity: Record Count (ACBISQ)

Except when the return ISNs option (I) is specified in Command Option 2, Adabas uses this field to return the number of records containing the value returned in the record buffer.

If the prefetch option is specified, the count of prefetched values is in the header that precedes the value in the ISN buffer.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be at least as large as the length specified.

Search Buffer Length (ACBSBL)

The search buffer length (in bytes). The search buffer area defined in the user program must be at least as large as the length specified.

Value Buffer Length (ACBVBL)

The value buffer length (in bytes). The value buffer area defined in the user program must be at least as large as the length specified.

ISN Buffer Length: Only with Command-Level Multifetch/Prefetch Option (ACBIBL)

The ISN buffer length (in bytes).

When the Command Option 1 field is set to "P", the L9 command uses the ISN buffer to hold prefetched descriptor values. The ISN buffer must be large enough to hold the largest descriptor value plus a 16-byte header preceding each value. The actual ISN buffer area defined in the user program must be at least as large as the length specified.

Command Option 1: Command-Level Multifetch/Prefetch Option (ACBCOP1)

Specifying one of these options indicates that the (command-level) prefetch or multifetch option is to be used. The multifetch/prefetch option can improve performance reading several descriptor values at a time thereby eliminating the time needed for single-record fetches. For more information, see the section *Using the Multifetch/Prefetch Feature*.

Option	Description
M (multifetching)	Multifetch processing is enabled. In conjunction with this option, you can set: ■ a non-zero value in the ISN Lower Limit field to limit the number of values to be multifetched. ■ zero in the ISN Lower Limit field to multifetch all values. The format of the ISN buffer data with the M option reflects the standard record descriptor data format.
P (prefetching)	Prefetch processing is enabled.

If the M or P options are specified, the L9 command puts all prefetched or multifetched descriptor values in the ISN buffer, preceded by a 16-byte header. The ISN buffer must be large enough to hold the largest descriptor value plus 16 bytes. The actual ISN buffer area defined in the user program must be at least as large as the ISN buffer length specified.

Command Option 2: Return ISNs Option (ACBCOP2)

Option	Description
I	Adabas stores the ISNs for each value in the record buffer, as well as the values themselves. The L9 command reads the Associator inverted lists only; no Data Storage access is required.
A	The descriptor's entries are processed in ascending order.
D	The descriptor's entries are processed in descending order.

Additions 1: Descriptor Name (ACBADD1)

If both the search and value buffer lengths are set to zero, a value in the Additions 1 field is the name of the descriptor for which values are to be returned. The name must be the same as the descriptor name specified in the format buffer.

In this case, L9 processes all values for the specified descriptor from the beginning of the file.

The descriptor name is specified in the first two positions of this field. The remaining positions must be set to blanks.

Additions 2: Response Subcodes (ACBADD2)

If the L9 command returns a nonzero response code, the rightmost two bytes of this field may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 5: Format ID, Global Format ID (ACBADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command ID, Format ID, Global Format ID*** for more information and examples.

ACB Examples

- Example 1
- Example 2
- Example 3: Overview of Sequence Options

Example 1

All values for the descriptor RB in file 2 are to be returned.

Control Block

Command Code	L9	
Command ID	L901	a non-blank CID is required
File Number	2	
Format Buffer Length	3	or larger
Record Buffer Length	10	or larger
Search Buffer Length	5	or larger
Value Buffer Length	1	or larger
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	RB.	the values are to be returned using standard length and format
Search Buffer	RB,1.	the values for descriptor RB are to be returned, and the starting value is being provided with standard format and length equal to 1
Value Buffer	b	processing is to begin with the first value for RB equal or greater than 'b'

Each successive L9 call will result in the return of the next value (values are provided in ascending order). The number of records containing the value is returned in the ISN quantity field.

Example 2

The values for descriptor AB in file 1 are to be returned. Only values which are equal to or greater than 20 are to be returned.

Control Block

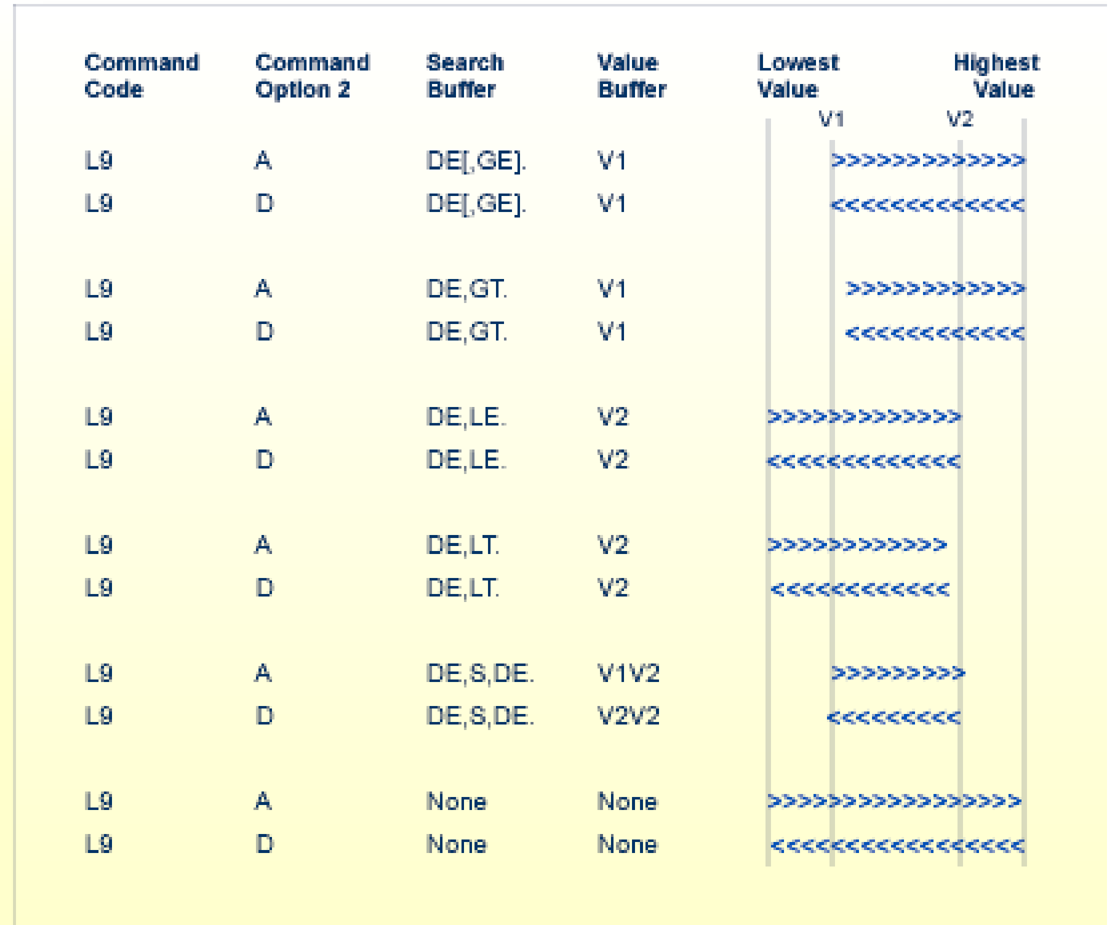
Command Code	L9	
Command ID	L902	a non-blank CID is required
File Number	1	
Format Buffer Length	7	or larger
Record Buffer Length	3	or larger
Search Buffer Length	7	or larger
Value Buffer Length	2	or larger
Additions 3	*bbbbbbbb* (blanks)	file 1 is not security-protected

Buffer Areas

Format Buffer	AB,3,U.	the values are to be returned with length=3 and with format=unpacked
Search Buffer	AB,2,U.	the values for the descriptor AB are to be returned, and the starting value is to be provided as a 2-byte unpacked number
Value Buffer	X'F2F0'	processing is to begin with the first value for AB that is equal to or greater than 20

Example 3: Overview of Sequence Options

The following table illustrates the possibilities for using the ascending/descending option in conjunction with various search buffer and value buffer contents. The following applies to the file being read:



Command Code	Command Option 2	Search Buffer	Value Buffer	Lowest Value V1 V2 Highest Value
L9	A	DE[,GE].	V1	>>>>>>>>>>>>>
L9	D	DE[,GE].	V1	<<<<<<<<<<<<<
L9	A	DE,GT.	V1	>>>>>>>>>>>>
L9	D	DE,GT.	V1	<<<<<<<<<<<<
L9	A	DE,LE.	V2	>>>>>>>>>>>>>
L9	D	DE,LE.	V2	<<<<<<<<<<<<<
L9	A	DE,LT.	V2	>>>>>>>>>>>>
L9	D	DE,LT.	V2	<<<<<<<<<<<<
L9	A	DE,S,DE.	V1V2	>>>>>>>>>
L9	D	DE,S,DE.	V2V2	<<<<<<<<<
L9	A	None	None	>>>>>>>>>>>>>>>>>
L9	D	None	None	<<<<<<<<<<<<<<<<<

Overview of Sequence Options

ACBX Interface Direct Call: L9 Command

This section describes ACBX interface direct calls for the L9 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	---	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	A
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
---	65-68	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
	77-84	---	---	---
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A

Field	Position	Format	Before Adabas Call	After Adabas Call
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	---	A
Search	F	U
Value	F	U
Multifetch *	---	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	The multifetch buffer is required only if the multifetch option is specified
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

L9

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Response code 3 (ADARSP003) indicates that an end-of-file condition has been detected.

Command ID (ACBXCID)

This field must be set to a non-blank, non-zero value. This value is used by Adabas to provide the values in the correct sequence and to avoid the repetitive interpretation of the format buffer.

This field must not be modified during a given sequential pass of the file.

The first byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN: Periodic Group Occurrence (ACBXISN)

If the descriptor for which values are to be returned is contained within a periodic group, Adabas returns in this field the occurrence number in which the value being returned is located. The occurrence number is provided in binary format in the two low-order bytes.

If the prefetch option is specified, any occurrence for prefetched values is returned in the header preceding the value in the ISN buffer.

ISN Lower Limit: Lowest ISN in Record Buffer (ACBXISL)

Adabas returns values in this field as follows:

Cmd Op 1	Cmd Op 2	The L9 command . . .
blank	I	returns the ISNs for each value in the record buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the returned ISN list in the ISN lower limit field.
P(refetch)	I	returns the first ISN in the record buffer and all prefetched descriptor values in the ISN buffer, preceded by a 16-byte header; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value prefetched in the ISN lower limit field.
M(ultifetch)	I	returns the group of multifetched records in the record buffer and a description of these records in the caller's ISN buffer; no value is returned in the ISN lower limit field.
	not I	places the first ISN of the last value multifetched in the ISN lower limit field.

If Command Option 1 is set to "M" (multifetch option), you can set:

- a non-zero value in the ISN lower limit field to limit the number of values to be multifetched.
- zero in the ISN lower limit field to multifetch all values.

Refer to the section ***Using the Multifetch/Prefetch Feature*** for more information.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

ISN Quantity: Record Count (ACBXISQ)

Except when the "return ISNs" (Command Option I) is specified, Adabas returns in this field the number of records containing the value returned in the record buffer.

If the prefetch option is specified, the count of prefetched values is in the header that precedes the value in the ISN buffer.

Command Option 1: Command-Level Multifetch/Prefetch Option (ACBXCOP1)

Specifying an M option in this field indicates that the (command-level) multifetch option is to be used. The multifetch option can improve performance reading several descriptor values at a time thereby eliminating the time needed for single-record fetches. For more information, see the section ***Using the Multifetch/Prefetch Feature***.

Option	Description
M (multifetching)	Multifetch processing is enabled. In conjunction with this option, you can set: ■ a non-zero value in the ISN lower limit field to limit the number of values to be multifetched. ■ zero in the ISN lower limit field to multifetch all values. The format of the ISN buffer data with the "M" option reflects the standard record descriptor data format.

If the M option is specified, the L9 command puts all multifetched descriptor values in the multifetch buffer, preceded by a 16-byte header. The multifetch buffer must be large enough to hold the largest descriptor value plus 16 bytes. The actual multifetch buffer area defined in the user program must be at least as large as the multifetch buffer length specified in the corresponding multifetch buffer ABD.

Command Option 2: Return ISNs Option (ACBXCOP2)

Option	Description
I	returns the ISNs for each value in the record buffer. The L9 command reads the Associator inverted lists only; no Data Storage access is required.
A	the descriptor's entries are processed in ascending order.
D	the descriptor's entries are processed in descending order.

Additions 1: Descriptor Name (ACBXADD1)

If both the search and value buffer lengths are set to zero, a value in the Additions 1 field is the name of the descriptor for which values are to be returned. The name must be the same as the descriptor name specified in the format buffer.

In this case, L9 processes all values for the specified descriptor from the beginning of the file.

The descriptor name is specified in the first two positions of this field. The remaining positions must be set to blanks.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 5: Format ID, Global Format ID (ACBXADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field is used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command, Format, and Global Format IDs*** for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers apply to the L9 command:

- Format Buffer
- Record Buffer
- Search Buffer
- Value Buffer
- ISN Buffer
- Multifetch Buffer

Format Buffer

Note: Commas separate elements in the format buffer; a period terminates the syntax statement. For more information about format buffer syntax, read ***Format Buffers***, elsewhere in this guide.

The format in which the values are to be returned must be specified in this buffer.

The syntax of the format buffer for L9 operation is:

```
name  [ ,length ]  [ ,format ] .
```

where:

name	The name of the descriptor for which values are to be returned. A phonetic descriptor may not be specified. A collation descriptor may only be specified if the decode option has been specified in its user exit: the value returned for it is not the index but the original field value. Subdescriptors, superdescriptors, hyperdescriptors and descriptors defined as a multiple-value field may be specified.
length	The length in which the value is to be returned. If length is not specified, the value will be returned using the standard length of the descriptor.
format	The format in which the value is to be returned. The format specified must be compatible with the standard format of the descriptor. If no format is specified, the value will be returned using the standard format of the descriptor.

Record Buffer

The descriptor value for the descriptor specified in the search and value buffers is returned in this field. A different value is provided with each L9 call. If the descriptor is defined with the null value suppression option, no value for the descriptor will be returned. If the Command Option 2 field is set to "I", Adabas returns a list of ISNs containing the requested value as well as the value itself in this buffer. The ISNs are provided in ascending sequence.

The descriptor value is provided as follows:

```
length  value  count  ISN-list
```

where:

`length`	A one-byte binary value which is the length of the value being provided. If the value has a standard length according to the descriptor type, this field is zero.
`value`	A descriptor value.
`count`	The number of values present in the file for the specified descriptor. This number may be returned in one or two bytes. If in one byte, the format is X'cc' where "cc" is the count; if in two bytes, the format is X'8ccc' where "ccc" is the count.
`ISN-list`	If the Command Option 2 field specified "I", the rest of the record buffer contains ISNs of the records containing this value. One record buffer will contain only the ISNs from one NI (normal index) block. Therefore: ■ the record buffer should be large enough to contain an entire NI block; and ■ the same value may appear several times with ascending ISNs.

Search Buffer

Note: Commas separate elements in the search buffer; a period terminates the syntax statement. For more information about search buffer syntax, read *Search Buffers*, elsewhere in this guide.

If both the search and value buffer lengths are set to zero, a value in the Additions 1 field is the name of the descriptor for which values are to be returned. In this case, L9 processes all values for the specified descriptor from the beginning of the file. The search and value buffers are not used.

If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

The length and format of the descriptor value as provided in the value buffer must be specified in the search buffer if different from the standard length and/or format of the named descriptor.

When a single value is provided in the value buffer (that is, the starting value when Command Option 2 is set to 'A' or the ending value when Command Option 2 is set to 'D'), the syntax for the search buffer is

```
name [ i ]  [ ,length ]  [ ,format ] [ ,comparator ].
```

When two values are provided in the value buffer, the syntax for the search buffer is

```
name [ i ]  [ ,length ] [ ,format ]  ,S  ,name  [ ,length ] [ ,format].
```

The elements used in the search buffer syntax statements are as follows:

name	The name of the descriptor for which values are to be returned. The name must be the same as that specified in the format buffer.
i	A one- to three-digit occurrence number subscript appended to the descriptor name if the descriptor is contained within a periodic group and only values within that particular occurrence are to be returned.
length	The length of the value provided in the value buffer. If the length is not specified, it is assumed that the value is being provided using the standard length of the descriptor.
format	The format of the value provided in the value buffer. If the format is not specified, it is assumed that the value is being provided using the standard format of the descriptor.
comparator	The scope of the read sequence: GE greater than or equal to the value to/from the highest value (the default) GT greater than the value to/from the highest value LE less than or equal to the value to/from the lowest value LT less than the value to/from the lowest value
S	A FROM-TO range (inclusive) that involves two search expressions. The same descriptor must be used in both expressions: AA,S,AA. valid AA,S,AB. invalid

Value Buffer

If both the search and value buffer lengths are set to binary zeros, a value in the Additions 1 field is the name of the descriptor for which values are to be returned. In this case, L9 processes all values for the specified descriptor in the sequence specified in the Command Option 2 field: from the beginning (A option) or end (D option) of the file. The search and value buffers are not used.

If a starting value, ending value, or both are specified in the value buffer, the search buffer is required in order to limit the number of descriptor entries retrieved.

If the search buffer comparator is GE or GT and a single value is provided in the value buffer, it represents the starting value when Command Option 2 is set to A or the ending value when Command Option 2 is set to D.

When two values are provided in the value buffer, the first specifies the lower limit of a range and the second specifies the upper limit. Each value is either the starting or ending value depending on the Command Option 2 setting of A or D.

If a value specified is not present in the file, Adabas finds the next higher or lower value, depending on the Command Option 2 setting of A or D. In this case, search buffer comparator LE is equivalent to LT and GE is equivalent to GT. If the value specified is not present and no higher (or lower) value is present, response code 3 (ADARSP003) is returned.

ISN Buffer

The ISN buffer is used only if the command-level multifetch option is used and if this command is issued using the **ACB direct call interface**. The information held in the ISN buffer following an L9 command results from specifying the M (multifetch) or P (prefetch) option in the Command Option 1 field. The format of the ISN buffer data depends on which option was specified.

- Data Format for the Multifetch Option (M)
- Data Format for the Prefetch Option (P)

Data Format for the Multifetch Option (M)

See section *READ (Lx) Multifetch Processing* for the record descriptor data format.

Data Format for the Prefetch Option (P)

Note: The prefetch option is valid only for **ACB interface direct calls**.

The ISN buffer holds the optionally prefetched descriptor values, each preceded by a 16-byte header. The 16-byte header preceding each value has the following format:

Byte	Usage
1-2	length of descriptor (including this header)
3-4	nucleus response code
5-8	nucleus internal ID
9-12	periodic group occurrence (see the ISN field description)
13-16	record count (see the ISN quantity field description)

Multifetch Buffer

The multifetch buffer is used only if the command-level multifetch option is used and if this command is issued using the **ACBX direct call interface**. See section *READ (Lx) Multifetch Processing* for the record descriptor data format. For more information, read *Using the Multifetch/Prefetch Feature*, elsewhere in this guide.

Additional Considerations

The following additional considerations are applicable for the L9 command:

1. The command ID used with the L9 command is saved internally and used by Adabas. It is released by Adabas when an end-of-file condition is detected, an RC or CL command is issued, or the Adabas session is terminated. Until it is released, the command ID may not be used for another command.
2. If another user is updating the file being read with an L9 command, it is possible that you will not receive one or more values in the file while reading with the L9 command.
3. Records can be updated in or deleted from a file being read by an L9 command. Adabas attempts to keep track of the last and the next value to be provided to the L9 command, and to provide the correct next value despite any interim update or deletion. However, if the record about to be accessed by the L9 command changes for some reason (it is updated or deleted by another user, for example), the L9 command continues processing as though no change occurred. In other words, a record deleted just before its inverted list entry is accessed by the L9 command is still considered a valid entry by the L9 command.
4. An internal format buffer used by an L9 command must have been created by a previous L9 command. Non-L9 commands cannot use internal format buffers created by L9 commands.

35 LF Command: Read Field Definitions

The LF command reads the characteristics of all fields in a file.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The LF command is used to read the field definition information for a file. This command is used primarily by Adabas subsystems; it is normally not used by an application program.

The user specifies the file number for which the field definitions are to be returned.

Adabas provides the field information in the record buffer in one of four formats, according to the setting of the Command Option 2 field. The timestamp when the field definitions were created or last updated is returned in the record buffer with the Command Option 2 field is set to X or F. If a file is stored with ADAORD or restored with ADASAV, the timestamp of the store or restore is not stored; the original timestamp of the file when it was reordered or saved is kept.

ACB Interface Direct Call: LF Command

This section describes ACB interface direct calls for the LF command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
	5-8	--	--	--
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-26	--	--	--
Record Buffer Length	27-28	binary	F	U
	29-35	--	--	--
Command Option 2	36	alphanumeric	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
	37-48	--	--	--
Additions 3	49-56	alphanumeric	F	A
	57-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	--
Record	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

LF

File Number (ACBFNR)

The number of the file for which the field definition information is to be returned.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If the LF command returns a nonzero response code, the rightmost two bytes of Adabas control block bytes 45-48 (Additions 2 field) may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The length specified must be large enough to contain all field definition information for the file, but not larger than the size of the record buffer area defined in the user program. If your FDT is larger than 32,767 bytes use an ACBX call (with its APLX/ADACBX interface) where larger buffers are supported.

Command Option 2: Type of Information to Be Displayed (ACBCOP2)

The setting of the Command Option 2 field determines the format and type of field information to be returned in the record buffer.

Option	Description
F	This option delivers the same information as the "X" option (below), including information on logically deleted fields and descriptors. Logically deleted fields will be returned with a specific indicator. All entries with logically deleted descriptors will be returned with a specific indicator.
I	Returns all field information in Adabas internal format.
S	Returns all field information, including collation descriptor, subfield and superfield, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor information. However, this information may not be complete; in particular, new features introduced beginning with Adabas Version 8.2 are not included. **Note:** Very large FDTs cannot be read with the ACB interface if the required space exceeds the record buffer size limit of 32 KB (minus 1 byte) length.
X	Returns extended field information, including collation descriptor, subfield, superfield, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor information with extended date and time formats. For logically deleted fields, no entry will be returned. For logically deleted descriptors on fields, subdescriptors, and superdescriptors, an entry will be returned with the descriptor options reset. For collation descriptors, hyperdescriptors, and phonetic descriptors, no entry will be returned. **Note:** Very large FDTs cannot be read with the ACB interface if the required space exceeds the record buffer size limit of 32 KB (minus 1 byte) length.

If this field is left blank or contains binary zero, the LF command returns field information *excluding* sub-/super-/hyper-/phonetic or collation descriptor information. This is the same format as provided in Adabas version 4.

Additions 3: Password (ACBADD3)

This field is used to provide a security password. If the file to be used is not security-protected, this field should be set to blanks. If the file is security-protected, the user must provide a valid password.

If the accessed file is password-protected, Adabas replaces the password with blanks during command processing to protect password integrity.

ACB Example

The field definition information for file 1 is to be read.

Control Block

Command Code	LF	
File Number	1	field definitions requested for file 1
Record Buffer Length	100	
Command Option 2	S	information for all types of descriptors and sub/superfields is to be returned
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected

ACBX Interface Direct Call: LF Command

This section describes ACBX interface direct calls for the LF command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
	13-16	---	---	---
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-49	---	---	---
Command Option 2	50	alphanumeric	F	U
	51-68	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A

Field	Position	Format	Before Adabas Call	After Adabas Call
	77-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	*	--
Record	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but should be included in Adabas call or one will be automatically generated.
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
LF

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file for which the field definition information is to be returned.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

Command Option 2: Type of Information to Be Displayed (ACBXCOP2)

The setting of the Command Option 2 field determines the format and type of field information to be returned in the record buffer.

Option	Description
F	his option delivers the same information as the "X" option (below), including information on logically deleted fields and descriptors. Logically deleted fields will be returned with a specific indicator. All entries with logically deleted descriptors will be returned with a specific indicator.
I	Returns all field information in Adabas internal format.
S	Returns all field information, including collation descriptor, subfield and superfield, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor information. **Note:** Beginning with Adabas 8.2, new field options (e.g. date-time and system fields) are not returned when Command Option 2 is set to "S", due to the fixed length of the field information output element. We recommend you set Command Option 2 to "X" instead.
X	Returns extended field information, including collation descriptor, subfield, superfield, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor information with extended date and time formats. For logically deleted fields, no entry will be returned. For logically deleted descriptors on fields, subdescriptors, and superdescriptors, an entry will be returned without descriptor options set. For collation descriptors, hyperdescriptors, and phonetic descriptors, no entry will be returned.

If this field is left blank or contains binary zero, the LF command returns field information *excluding* sub-/super-/hyper-/phonetic or collation descriptor information. This is the same format as provided in Adabas version 4.

Additions 3: Password (ACBXADD3)

This field is used to provide a security password. If the file to be used is not security-protected, this field should be set to blanks. If the file is security-protected, the user must provide a valid password.

If the accessed file is password-protected, Adabas replaces the password with blanks during command processing to protect password integrity.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

A format buffer is not used by the LF command, but should be included in the Adabas call.

All field definition information is returned in the record buffer, in one of four formats, depending on the setting of the Command Option 2 field.

This section covers the following topics:

- Record Buffer When Command Option 2 Not Set

Format Buffer

A format buffer is not used by the LF command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer When Command Option 2=S

If the Command Option 2 field is set to "S", all field information, including collation descriptor, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor and sub-/superfield information is returned in the following format:

Note: If you are using Adabas 8, and an LF command with Command Option 2 set to "S" is run, and large object (LB) fields are encountered, the LB field description is returned in an F-type FDT field definition. Bit 6 in the second format byte (at offset 7 or byte 8 in the element) is set to indicate that the LB (large object) option is set for the field. In addition, bit 1 of the second format byte indicates whether the LB field is defined with the NB (no blank compression) option.

Bytes	Usage
1-2	total length of information
3-4	number of fields in the FDT (including SDTs, as described below)
5-N	field entries; each entry is 8 bytes, maximum number of entries is 3214
(N+1) - M	special descriptor table (SDT) including ▪ sub-/superdescriptors (or sub-/superfields) ▪ phonetic descriptors ▪ hyperdescriptors ▪ collation descriptors The length of a sub-/phonetic/collation descriptor element is eight bytes. Superdescriptor or hyperdescriptor elements are two or more 8-byte entries long.

The following sections provide details of this format:

- FDT Field Entries
- SDT Field Definitions
- Collation Descriptor Entries
- Hyperdescriptor Entries
- Phonetic Descriptor Entries
- Subfield/Subdescriptor Entries

- Superdescriptor/Superfield Entries

FDT Field Entries

The syntax of FDT field entry is:

```
'F' field-name option level length format
```

The following table describes this syntax:

Notation	Bytes	Usage
F	1	"F" indicates an FDT field entry
field-name	2-3	field name
option	4	definition options: ▪ bit 1=1: descriptor ▪ bit 2=1: fixed length ▪ bit 3=1: multiple-value field ▪ bit 4=1: null suppression ▪ bit 5=1: periodic group field ▪ bit 6=1: parent of phonetic descriptor ▪ bit 7=1: parent of subdescriptor or superdescriptor ▪ bit 8=1: unique descriptor
level	5	level number (in binary)
length	6	length
format	7	type of data: ▪ A -- alphanumeric ▪ B -- binary ▪ F -- fixed point ▪ G -- floating point ▪ P -- packed decimal ▪ U -- unpaced decimal ▪ W -- wide-character

Notation	Bytes	Usage
	8	options (continued): - bit 1=1: NB (no blank compression) option - bit 2=1: NV (not converted) option - bit 3=1: unused - bit 4=1: XI (exclude PE occurrence number from UQ) option - bit 5=1: LA (long alpha) option - bit 6=1: LB (large object) option - bit 7=1: NN option - bit 8=1: NC option

Note: A field within a periodic group has the following characteristics: - an option field (byte 4) with bit 5=1; and - a level field (byte 5) with level number *greater than* 1. The periodic group field itself always has option bit 5=1 and a level number of 1.

SDT Field Definitions

```
x   SDT-definition
```

where *x* is one of the following:

Value of *x*	Description
C	Collation descriptor; see the section *Collation Descriptor Entries*
H	Hyperdescriptor; see the section *Hyperdescriptor Entries*
P	Phonetic descriptor; see the section *Phonetic Descriptor Entries*
S	Subfield or subdescriptor;, see the section *Subfield/Subdescriptor Entries*
T	Superfield or superdescriptor; see the section*Superdescriptor/Superfield Entries*
X'00'	Element continuation

Collation Descriptor Entries

```
'C' name option exit length p-field-name
```

Notation	Bytes	Usage
'C'	1	'C' indicates collation descriptor
name	2-3	collation descriptor name
option	4	definition options: bit 1=1 descriptor bit 2=1 exclude PE occurrence number from UQ bit 3=1 multiple-value format bit 4=1 null-value suppression bit 5=1 periodic-group field bits 6-7 (unused) bit 8=1 unique descriptor
exit	5	collation exit number (binary, values 1-8 permitted)
length	6	length
p-field-name	7-8	parent-field name

Hyperdescriptor Entries

```
'H' name option exit length format X'00'
X'00' X'00' p-fieldname-list ...
```

Notation	Bytes	Usage
'H'	1	'H' indicates a hyperdescriptor definition
name	2-3	hyperdescriptor name
option	4	definition options: bit 1 (unused) bit 2=1 fixed length bit 3=1 multiple value bit 4=1 null-value suppression bit 5=1 periodic group bit 6-7 (unused) bit 8=1 unique descriptor
level	5	hyperdescriptor exit number (binary; values 1-31 permitted)
length	6	length
format	7	format:

Notation	Bytes	Usage
		A alphanumeric B binary F fixed point G floating point P packed decimal U unpacked decimal
X'00'	8	options (continued)
		bits 1-3 (unused) bit 4=1 XI (exclude PE occurrence number from UQ) option bits 5-8 (unused)

A hyperdescriptor parent-field name list is an extension of a hyperdescriptor definition. It has the following format for all eight-byte groups after the first:

Notation	Bytes	Explanation
X'00'	1	X'00' indicates continuation
X'00'	2	(unused)
p-fieldname-list...	3-8	parent-field name list: each name is two bytes; six bytes total (that is, three names. If fewer than three names are provided, the additional bytes are filled with X'00').

Phonetic Descriptor Entries

```
'P' desc-name option p-field-name X'0000'
```

Notation	Bytes	Explanation
'P'	1	'P' indicates phonetic descriptor
desc-name	2-3	phonetic descriptor name
option	4	(unused)
p-field-name	5-6	parent-field name
X'0000'	7-8	(unused; set to nulls)

Subfield/Subdescriptor Entries

```
'S' s-name option p-field-name from to
```

Notation	Bytes	Usage
'S'	1	'S' indicates subdescriptor/subfield
s-name	2-3	subdescriptor or subfield name
option	4	definition options:
		bit 1=1 descriptor
		bit 2=1 exclude PE occurrence number from UQ
		bit 3=1 multiple-value format
		bit 4=1 null-value suppression
		bit 5=1 periodic-group field
		bit 6-7 (unused)
		bit 8=1 unique descriptor
p-field-name	5-6	parent-field name
from	7	starting (inclusive) byte
to	8	ending (inclusive) byte

Superdescriptor/Superfield Entries

```
'T' sup-name option p-field-name from to
X'00' X'000000'p-field-name from to
```

Notation	Bytes	Usage
'T'	1	'T' indicates superdescriptor/superfield
sup-name	2-3	superdescriptor name
option	4	definition options:
		bit 1=1 descriptor
		bit 2=1 exclude PE occurrence number from UQ
		bit 3=1 multiple-value format
		bit 4=1 null-value suppression
		bit 5=1 periodic-group field
		bit 6-7 (unused)
		bit 8=1 unique descriptor
p-field-name	5-6	parent-field name
from	7	starting (inclusive) byte

Notation	Bytes	Usage
to	8	ending (inclusive) byte

Extension of a superdescriptor or superfield definition has the following format on all eight-byte groups after the first:

Notation	Bytes	Explanation
X'00'	1	indicates continuation
X'000000'	2-4	(unused)
p-field-name	5-6	parent-field name
from	7	starting (inclusive) byte
to	8	ending (inclusive) byte

Record Buffer When Command Option 2=F or X

If the Command Option 2 field is set to "F" or "X", extended field information, including collation descriptor, subdescriptor, superdescriptor, hyperdescriptor, and phonetic descriptor and sub-/superfield information is returned in the following format:

The "F" option delivers the same information as the "X" option, including information on logically deleted fields and descriptors. Logically deleted fields will be returned with a specific indicator; entries with logically deleted descriptors will be returned with the specific indicator.

Bytes	Usage
1-4	Total length of information
5	Structure level
6	Flag byte for future use
7-8	Number of field definition entries in the FDT (including SDTs, as described below). The maximum number of entries is 3214.
9-16	UNIX timestamp (the number of microseconds since 1970)
17-N	FDT Field entries: each entry is 16 bytes long. Future versions of Adabas may introduce additional, larger entries. Therefore, unknown entry types should not be considered in error.
(N+1) - M	Special descriptor table (SDT) entries including: ■ sub-/superdescriptors (or sub-/superfields) ■ phonetic descriptors ■ hyperdescriptors ■ collation descriptors For SDT entries, an SDT element has an integral length and is a multiple of 8 bytes.
(M+1) - L	Referential integrity constraint table (RIT). This table is not applicable to mainframe systems; it is used in Adabas open systems environments only.

The following sections provide format details of the field and descriptor entries included in the record buffer:

- FDT Field Entries
- SDT Field Entries
- Collation Descriptor Entries
- Hyperdescriptor Entries
- Phonetic Descriptor Entries
- Subfield/Subdescriptor Entries
- Superdescriptor/Superfield Entries

These entries may be larger with each Adabas version. Therefore, you should always use the entry field length (in byte 2) to skip to the next entry. The length of each entry is aligned to four bytes so you can access 4-byte integer values without alignment problems.

FDT Field Entries

The following table describes the FDT field entry format when Command Option 2 is set to "F" or "X".

Bytes	Usage
1	"F" indicates an FDT field definition
2	Total length of FDT field entry
3-4	Field name
5	Field format: ▪ A: alphanumeric ▪ B: binary ▪ F: fixed point ▪ G: floating point ▪ P: packed decimal ▪ U: unpaced decimal ▪ W: wide-character
6	Definition options: ▪ bit 1=1: descriptor ▪ bit 2=1: fixed length ▪ bit 3=1: multiple-value field ▪ bit 4=1: null suppression ▪ bit 5=1: periodic group field ▪ bit 6=1: parent of phonetic descriptor

Bytes	Usage
	■ bit 7=1: parent of subdescriptor or superdescriptor ■ bit 8=1: unique descriptor
7	Additional options: ■ bit 1=1: NB (no blank compression) option ■ bit 2=1: NV (not converted) option ■ bit 3=1: unused ■ bit 4=1: XI (exclude PE occurrence number from UQ) option ■ bit 5=1: LA (long alpha) option ■ bit 6=1: LB (large object) option ■ bit 7=1: NN option ■ bit 8=1: NC option
8	Level number (in binary)
9	Date/time edit masks: ■ 1: E(DATE) ■ 2: E(TIME) ■ 3: E(DATETIME) ■ 4: E(TIMESTAMP) ■ 5: E(NATDATE) ■ 6: E(NATTIME) ■ 7: E(UNIXTIME) ■ 8: E(XTIMESTAMP)
10	For the datetime/edit masks, the following additional information is supplied: ■ bit 0x01: TZ (timezone) option For the SY option, the following additional information is supplied: ■ bit 0x40: CR option in use
11	SY function options or the user exit number: ■ 1: TIME ■ 2: SESSIONID ■ 3: OPUSER ■ 4: SESSIONUSER (mainframe only) ■ 5: JOBNAME (mainframe only)

Bytes	Usage
12	Additional field status information*: - bit 0x01: Field logically deleted - bit 0x02: Descriptor logically deleted (mainframe only) *: This information will only be given when Command Option 2 is set to "F"; other descriptor-related flags also remain set. When Command Option 2 is set to "X", the field entry for the logically deleted field will be suppressed entirely; descriptor-related flags for logically deleted descriptors will be reset.
13-16	Field length

Note: A field within a periodic group has the following characteristics: - an option field (byte 6) with bit 5=1; and - a level field (byte 8) with level number *greater than* 1. The periodic group field itself always has option bit 5=1 and a level number of 1.

SDT Field Entries

```
x   SDT-definition
```

where *x* is one of the following:

Value of *x*	Description
C	Collation descriptor entry; see the section *Collation Descriptor Entries*
H	Hyperdescriptor entry; see the section *Hyperdescriptor Entries*
P	Phonetic descriptor entry; see the section *Phonetic Descriptor Entries*
S	Subfield or subdescriptor entry; see the section *Subfield/Subdescriptor Entries*
T	Superfield or superdescriptor entry; see the section*Superdescriptor/Superfield Entries*
X'00'	Element continuation

Collation Descriptor Entries

The following table describes the format of collation descriptor entries:

Bytes	Usage
1	"C" indicates a collation descriptor
2	Total length of collation descriptor entry.
3-4	Collation descriptor name
5	Parent field format (only "W" is possible in Adabas open systems environments; both "A" and "W" are possible in Adabas mainframe environments).

Bytes	Usage
6	Definition options: ■ bit 1=1: descriptor ■ bit 2=1: XI (exclude PE occurrence number from UQ) option ■ bit 3=1: multiple-value field ■ bit 4=1: null suppression ■ bit 5=1: periodic group field ■ bit 6=1: parent of phonetic descriptor ■ bit 7=1: parent of subdescriptor or superdescriptor ■ bit 8=1: unique descriptor
7-8	Standard length
9-10	Parent field name
11-12	Maximum internal length
13	Additional options: ■ 0x01: NC (open systems) ■ 0x02: Descriptor logically deleted (mainframe systems only)* ■ 0x04: LA (open systems only) ■ 0x08: LB (open systems only) ■ 0x80: Collation defined via collation exit When this bit is not set, the collation is defined via ICU ■ Other bits are unused *: This information will only be given when Command Option 2 is set to "F"; descriptor-related flags also remain set. When Command Option 2 is set to "X", the entry for the logically deleted descriptor will be suppressed entirely.
14	String length of the collation attribute string (length byte and termination null character not included)
15-(14+byte14)	Collation attribute string as null terminated string. For example: `"'de',PRIMARY"` If the collation is defined via the collation exit, the exit number as a null terminated string is placed here. For example: `"1"`.

Notes:

1. The collation descriptor entry length is aligned to four bytes.
2. The format of the parent field is relevant if you specify a parent field value in the value buffer. In Adabas on open systems, you can also specify the internal collation descriptor values if the

collation descriptor is defined without the HE option. These values are values of format A with option NV.

Hyperdescriptor Entries

The following table describes the format of hyperdescriptor entries:

Bytes	Usage
1	"H" indicates a hyperdescriptor definition
2	Total length of the hyperdescriptor entry
3-4	Hyperdescriptor name
5	Field format
6	Definition options: ■ bit 1=1: descriptor ■ bit 3=1: multiple-value field ■ bit 4=1: null suppression ■ bit 5=1: periodic group field ■ bit 6=1: parent of phonetic descriptor ■ bit 7=1: parent of subdescriptor or superdescriptor ■ bit 8=1: unique descriptor
7-8	Hyperdescriptor length
9	Exit number
10	Additional field status information:* ■ bit 0x02: Descriptor logically deleted (mainframe only) *: This information will only be given when Command Option 2 is set to "F"; descriptor-related flags remain set. When Command Option 2 is set to "X", the field entry for the logically deleted field will be suppressed entirely.
11	Unused
12	Number of parent fields
13-(12+2* *byte-12*)	Parent field names

Note: The hyperdescriptor entry length is aligned to four bytes.

Phonetic Descriptor Entries

The following table describes the format of phonetic descriptor entries:

Bytes	Usage
1	"P " identifies a phonetic descriptor
2	Total length of phonetic descriptor entry
3-4	Phonetic descriptor name
5	Field format (currently only "A"-format supported)
6	Additional field status information:* *: This information will only be given when Command Option 2 is set to "F". When Command Option 2 is set to "X", the field entry for the logically deleted field will be suppressed entirely.
7-8	Phonetic descriptor length
9-10	unused
11-12	Phonetic descriptor parent field name

Subfield/Subdescriptor Entries

The following table describes the format of subfield or subdescriptor entries:

Bytes	Usage
1	"S" identifies a subdescriptor or subfield
2	Total length of the subfield or subdescriptor entry.
3-4	Subdescriptor or subfield name
5	Field format
6	Definition options: ■ bit 1=1: descriptor ■ bit 2=1: XI (exclude PE occurrence number from UQ) option ■ bit 3=1: multiple-value field ■ bit 4=1: null suppression ■ bit 5=1: periodic group field ■ bit 6=1: parent of phonetic descriptor ■ bit 7=1: parent of subdescriptor or superdescriptor ■ bit 8=1: unique descriptor
7-8	Descriptor length
9	Additional field status information: * ■ bit 0x02: Descriptor logically deleted (mainframe systems only)

Bytes	Usage
	*: This information will only be given when Command Option 2 is set to "F"; descriptor-related flags also remain set. When Command Option 2 is set to "X", the entry for the logically deleted descriptor will appear with all descriptor related flags reset.
10	Number of parent fields ("1" for every subfield)
11-(10+ 6 * *byte10*)	Parent field entries. For each parent field, the following format is used: ■ Parent field name (2 bytes) ■ From byte (2 bytes) ■ To byte (2 bytes)

Note: The subdescriptor entry length is aligned to four bytes.

Superdescriptor/Superfield Entries

The following table describes the format of superfield or superdescriptor entries:

Bytes	Usage
1	"T" identifies a superdescriptor or superfield
2	Total length of the superfield or superdescriptor entry.
3-4	Superdescriptor or superfield name
5	Field format
6	Definition options: ■ bit 1=1: descriptor ■ bit 2=1: XI (exclude PE occurrence number from UQ) option ■ bit 3=1: multiple-value field ■ bit 4=1: null suppression ■ bit 5=1: periodic group field ■ bit 6=1: parent of phonetic descriptor ■ bit 7=1: parent of subdescriptor or superdescriptor ■ bit 8=1: unique descriptor
7-8	Descriptor length
9	Additional field status information: * ■ bit 0x02: Descriptor logically deleted (mainframe systems only) *: This information will only be given when Command Option 2 is set to "F"; descriptor-related flags also remain set. When Command Option 2 is set to "X", the entry for the logically deleted descriptor will appear with all descriptor related flags reset.

Bytes	Usage
10	Number of parent fields ("1" for every superdescriptor)
11-(10+ 6 * *byte10*)	Parent field entries. For each parent field, the following format is used: - Parent field name (2 bytes) - From byte (2 bytes) - To byte (2 bytes)

Record Buffer When Command Option 2=I

If "I" is set for Command Option 2, field information is returned in Adabas internal format:

Bytes	Contents
1	X'80'
2	B'00000*xyz*' where *xyz* are ciphering bits: 1=yes; 0=no
	x user y new z old
3-4	binary zeros
5-8	total inclusive length of FDT, including field definition table (FDT proper), FDT index, and special descriptor table (SDT) [= p – 4]
9-12	Inclusive length of field definition table (FDT proper) [= n – 8]
13-n	FDT field descriptor elements (20 bytes per element; see the following description)
n+1 to n+4	Inclusive length of FDT index [= m – n]
n+5 to m	FDT index
m+1 to m+4	Inclusive length of special descriptor table (SDT) [= p – m]
m+5 to p	Special descriptor table (SDT)

The format of each FDT field descriptor elements is described in the following table:

Offset	Contents
0	field level
1 - 2	field name
3	special field options
4 - 6	reserved
7	default field length
8	field format
9	descriptor definition options

Offset	Contents
10	special descriptor parent options
11	periodic group count field
12-13	FDT element chain pointer
14	field security levels
15 - 19	reserved

The meaning of FDT elements is described in the Adabas architecture training information.

Record Buffer When Command Option 2 Not Set

Note: Command option 2 may be set to values other than "I" or "S" to support older programs; these values do not support newer features. Software AG recommends "I" or "S" for new programs.

If the Command Option 2 field contains neither "I" nor "S", the field information returned *excludes* collation descriptor and sub-/super-/hyper-/phonetic descriptor information. The information is provided in the same format as provided in Adabas version 4:

```
n field-def
```

where:

n is the number of fields in the file. The number is provided as a four-byte binary number in the first four bytes of the record buffer.

field-def is the field definition information for each field within the file. The information for each field is provided in six bytes according to the following format:

<table>
<tr><th>Bytes</th><th>Usage</th></tr>
<tr><td>1</td><td>level number (binary)</td></tr>
<tr><td>2 - 3</td><td>name (alphanumeric)</td></tr>
<tr><td>4</td><td>standard length (binary)</td></tr>
<tr><td>5</td><td>standard format (alphanumeric):</td></tr>
<tr><td></td><td><table>
<tr><td>A</td><td>alphanumeric</td></tr>
<tr><td>B</td><td>binary</td></tr>
<tr><td>F</td><td>fixed point</td></tr>
<tr><td>G</td><td>floating point</td></tr>
<tr><td>P</td><td>packed decimal</td></tr>
<tr><td>U</td><td>unpacked decimal</td></tr>
<tr><td>W</td><td>wide-character</td></tr>
</table></td></tr>
<tr><td>6</td><td>definition options:</td></tr>
</table>

Bytes	Usage	
	bit1=1	descriptor
	bit 2=1	fixed storage
	bit 3=1	multiple-value field
	bit 4=1	null-value suppression
	bit 5=1	periodic group field
	bit 6=1	phonetic source field
	bit 7=1	sub-/superdescriptor source field
	bit 8=1	unique descriptor

The information for the next field immediately follows the information for the preceding field with no intervening spaces.

36 N1 and N2 Commands: Adding Records

The N1 and N2 commands are used to add a new record to a file.

The N1 command adds a new database record with an ISN assigned by Adabas. The N2 command adds a new database record with an ISN assigned by the user.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The user specifies the file to which the record is to be added, and the fields for which values are being provided. Any fields not specified will contain a null value in the record added.

Adabas assigns the record an ISN, adds the record to Data Storage, and performs any Associator updating which may be required.

The N2 command is used if the ISN to be assigned to the record is being provided by the user. To keep the ISN assigned to the same record, the unloaded file must be reloaded with the USER-ISN=YES option.

If the user is an ET logic user and is operating in multiuser mode, the record added is placed in hold status.

ACB Interface Direct Call: N1 and N2 Commands

This section describes ACB interface direct calls for N1 and N2 commands. It covers the following topics:

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	A/U[1]
ISN Lower Limit	17-20	binary	--	A[2]
ISN Quantity	21-24	binary	--	A[2]
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
	29-44	--	--	--
Additions 2	45-48	alphanumeric	--	A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Notes

1. Supplied by Adabas for N1; unchanged for N2.
2. These fields are used and not reset by Adabas if coupled files are used.

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

N1 or N2

Command ID (ACBCID)

If a series of records is being added using a series of N1 or N2 calls, and the same fields are specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction in the time required to process each N1 or N2 call.

If only a single record is being added, or if the format buffer is modified between N1 or N2 calls, this field should be set to blanks.

The first byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

Specify the binary number of the file to be read in this field. For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

If the N1 command is being executed, Adabas returns the ISN assigned to the record in this field.

If the N2 command is being executed, the ISN to be assigned to the record must be provided in this field. The ISN provided must not already be assigned to a record in the file, and must be within the limit (MAXISN) in effect for the file. MAXISN is set by the DBA when the file is loaded.

Note: You cannot assign an ISN that is greater than the value specified by the MAXISN parameter for the file.

ISN Quantity/Lower Limit (ACBISQ and ACBISL)

These fields are set to nulls following completion of the N1 or N2 command operation, unless hard-coupled files are used. If coupled files are used, these fields are used by N1 or N2 command processing and are not reset.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Additions 2: Length of Compressed Record (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the newly added compressed record;
- If the N1 or N2 command returns a nonzero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section *Command, Format, and Global Format IDs* for more information and examples.

ACB Examples

- Example 1
- Example 2

Example 1

A record is to be added to file 1. The ISN of the record is to be assigned by Adabas. The field values which are to be provided are as follows:

Field	Value
AA	ABCD
MF (value 1)	AAA
MF (value 2)	BBB
BA (1st occurrence)	5
BA (2nd occurrence)	6

Control Block

Command Code	N1	
Command ID	*bbbb* (blanks)	only 1 record being added
File Number	1	
Format Buffer Length	15	or larger
Record Buffer Length	16	or larger
Additions 3	*bbbbbbbb* (blanks)	file 1 is not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	AA,MF1-2,BA1-2.
Record Buffer	X'C1C2C3C440404040C1C1C1C2C2C20506'

Example 2

A record is to be added to file 2. The ISN of the record is to be provided by the user. The field values to be provided are as follows:

Field	Value
RA	12345678
RB	ABCD

Control Block

Command Code	N2	
Command ID	*bbbb* (blanks)	only 1 record is to be added
File Number	2	
ISN	20	ISN 20 is to be assigned to the record
Format Buffer Length	6	or larger
Record Buffer Length	18	or larger
Additions 3	password	file 2 is security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	RA,RB.
Record Buffer	X'F1F2F3F4F5F6F7F8C1C2C3C4404040404040'

ACBX Interface Direct Call: N1 and N2 Commands

This section describes ACBX interface direct calls for N1 and N2 commands. It covers the following topics:

- Control Block and Buffer Overview

- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	A/U[1]
	33-36	---	---	---
ISN Lower Limit	37-40	binary	---	A[2]
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A[2]
	49-64	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

Notes

1. Supplied by Adabas for N1; unchanged for N2.
2. These fields are used and not reset by Adabas if coupled files are used.

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	U

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
N1 or N2

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)
If a series of records is being added using a series of N1 or N2 calls, and the same fields are specified in the format buffer for each call, this field should be set to a non-blank, non-zero value. This results in a reduction in the time required to process each N1 or N2 call.

If only a single record is being added, or if the format buffer is modified between N1 or N2 calls, this field should be set to blanks.

The first byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

If the N1 command is being executed, Adabas returns the ISN assigned to the record in this field.

If the N2 command is being executed, the ISN to be assigned to the record must be provided in this field. The ISN provided must not already be assigned to a record in the file, and must be within the limit (MAXISN) in effect for the file. MAXISN is set by the DBA when the file is loaded.

Note: You cannot assign an ISN that is greater than the value specified by the MAXISN parameter for the file.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit (ACBXISL)

This field is set to nulls following completion of the N1 or N2 command operation, unless hard-coupled files are used. If coupled files are used, this field is used by N1 or N2 command processing and is not reset.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

ISN Quantity (ACBXISQ)

This field is set to nulls following completion of the N1 or N2 command operation, unless hard-coupled files are used. If coupled files are used, this field is used by N1 or N2 command processing and is not reset.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Format ID, Global Format ID (ACBXADD5)

Use this field to specify a separate format ID that identifies the internal format buffer used for this command, or to provide a global format ID allowing use of the internal format buffer by all users.

As long as the leftmost bit of the Additions 5 field is set to 0, the value provided in the command ID field will be used as the format ID as well.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

See the section ***Command, Format, and Global Format IDs*** for more information and examples.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

- Format Buffer

- Record Buffer

Format Buffer

The fields for which values are being provided in the record buffer must be specified in this buffer. When performing an N1 command, the format buffer cannot contain any of the following:

- format selection criteria ("field-name operator value..");
- an edit mask element;
- a reference to a sub-/superdescriptor field;
- the same field specified more than once (except a multiple-value field);
- an "-N" type of record format specification (for example, ABN or AB1 - N).

Any of the above in the format buffer cause a nucleus response of 44 for an N1 command. Any fields that are not specified in the format buffer will contain a null value in the record being added.

The following rules control the processing of non-NU descriptors which are not specified in the format buffer for an N1 or N2 command:

- Any omitted non-NU descriptor whose definition in the field definition table (FDT) is both the farthest from the beginning of the FDT, and *precedes* the FDT definition of the last field specified in the format buffer *will* have null values entered in the inverted list for the descriptor;
- Any omitted non-NU descriptor whose definition in the field definition table (FDT) is both the farthest from the beginning of the FDT, and *follows* the FDT definition of the last field specified in the format buffer will *not* have null values entered in the inverted list for the descriptor.

Therefore, the format buffer entry should reference either all non-NU descriptors, or at least one field following (in FDT order) all non-NU descriptor fields. This ensures that null values are correctly inserted in the inverted lists for all non-NU descriptors.

For non-NU descriptors that are contained in a periodic group, null values are entered in their inverted lists only for null occurrences that precede the highest occurrence number specified in the format buffer.

The following additional format buffer considerations are applicable for the N1 or N2 command:

1. Subdescriptors, superdescriptors, hyperdescriptors, and phonetic descriptors may not be specified in the format buffer. Adabas automatically creates the correct value for any of the above if a field from which such a descriptor is derived is specified in the format buffer.
2. Theoretically, the maximum record length permitted is 32767 bytes before compression. The actual maximum is limited by block size restrictions. It is also smaller depending on the size of the LU parameter specified for the Adabas session; the maximum is (LU - format buffer length - 108). The maximum record length after compression is equal to the smaller of either the Data Storage block size - 4 bytes, or the Work block size - 110 bytes.

3. If a field is specified using a length override that exceeds the standard length (not permitted if the field is defined with the fixed storage option), all subsequent references to this field should specify the length that was used. If a subsequent reference uses the standard length, value truncation for alphanumeric fields or a non-zero response code for numeric fields may occur.
4. Only a multiple-value field may be specified more than once in the format buffer.
5. A multiple-value count field, periodic group count field, or literal value specified in the format buffer is ignored by Adabas. The corresponding value in the record buffer is also ignored.
6. If a multiple-value field is specified in the format buffer, Adabas sets the multiple-value field count according to the following rules:

   - For a multiple-value field defined with the NU option, the count field is adjusted to reflect the number of existing nonblank values. Blank values are completely suppressed.

Field definition	01,MF,5,A,MU,NU
Format buffer	MF1-3.
Record buffer	XXXXXYYYYYZZZZZ
Result after add	XXXXX,YYYYY,ZZZZZ MF count = 3
Format buffer	MF1-3.
Record buffer	XXXXX*bbbbb*ZZZZZ
Result after add	XXXXX,ZZZZZ MF count = 2
Format buffer	MF1-3.
Record buffer	*bbbbbbbbbbbbbbb* (blanks)
Result after add	values suppressed MF count = 0

   - For a multiple value field defined without the NU option, the count is adjusted to reflect the number of existing values (including null values).

Field definition	01,MF,5,A,MU
Format buffer	MF1-3.
Record buffer	XXXXXYYYYY*bbbbb*
Result after add	XXXXX,YYYYY,b(blank) MF count = 3
Format buffer	MF1.
Record buffer	*bbbbb* (blanks)
Result after add	b (blank) MF count = 1

Up to 191 values are permitted for a multiple-value field.

7. If a periodic group or a field within a periodic group is specified in the format buffer, Adabas sets the periodic group count equal to the highest occurrence number specified in the format buffer. If the highest occurrence suppresses null values, the count is adjusted accordingly.

Field definitions	01,GB,PE 02,BA,1,B,DE,NU 02,BB,5,P,NU
Format buffer	GB1-2.
Record buffer	X'08000000500F09000000600F'
Result after add	GB (1st occurrence) BA = 8 BB = 500 GB (2nd occurrence) BA = 9 BB = 600 GB count = 2
Format buffer	GB1-2.
Record buffer	X'00000000000F00000000000F'
Result after add	GB (1st occurrence) values suppressed GB (2nd occurrence) values suppressed GB count = 0

Up to 191 occurrences are permitted for a periodic group.

8. 8. If a field defined with variable length (no standard length) is specified in the format buffer, the corresponding value in the record buffer must be preceded by a 1-byte binary number that represents the length of the value (including the length byte).

Field definitions	01,AA,3,A 01,AB,A
Format buffer	AA,AB.
Record buffer	X'F1F2F306F1F2F3F4F5'

Fields AA and AB are to be added. The value for AA is "123". The value for AB (which is a variable length field) is "12345".

Record Buffer

The value for each field specified in the format buffer must be provided in the record buffer.

Each value must be provided according to the standard length and format of the field for which the value is being provided, unless a different length and/or format is specified in the format buffer.

If the field is defined as a variable-length field (no standard length), a 1-byte binary field containing the length of the field (including the length byte) must be provided immediately before the value.

If the field for which the value is being provided is defined as a unique descriptor, the value provided must not already exist for the descriptor; otherwise, the command will be rejected.

37 OP Command: Open User Session

The OP command indicates the beginning of a user session and specifies options for that user session.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

Software AG recommends that all users start their Adabas sessions with an OP command.

An OP command is mandatory if any of the following is true for the user:

- The nucleus is run with ADARUN parameter OPENRQ=YES.
- Exclusive file control (EXF) is to be performed.
- User data that was stored in an Adabas system file by a previous ET command is to be read.
- User data is to be stored in an Adabas system file, using a C3, CL, or ET command.
- The user is to be assigned a special processing priority.
- The user is to be an access-only user (no update commands permitted).
- A transaction time limit or a non-activity time limit is to be set for the user that differs from that specified by ADARUN parameters TT or TNAx, respectively. The setting for a user must conform to the maximum (two-byte) setting set by the ADARUN parameters MXTT and MXTNA, respectively.
- Special data encoding or architecture is to be specified for the user session.

An OP command is otherwise optional. An implicit OP command is issued by Adabas when the first Adabas command is issued by a user who is not currently identified to Adabas.

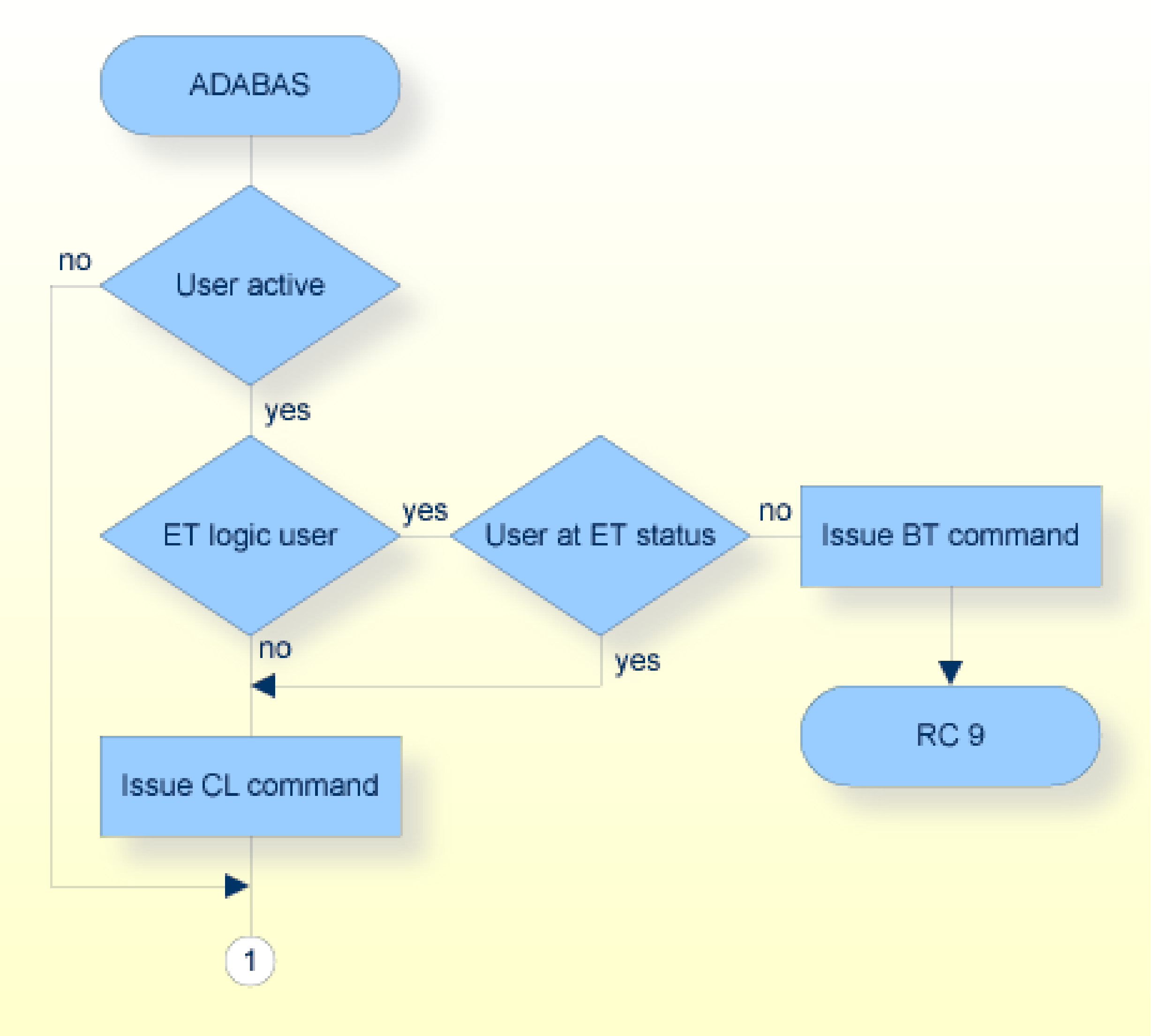
Users accessing files that are protected by Adabas Security are not required to issue an OP command; such users must, however, provide a password with each command directed to a security-protected file. Ask your DBA or system security specialist for more information.

If an OP command is issued by an *active* ET logic user, and the user is not at ET status (an OP, ET, or BT command has not been previously issued with one or more records in hold status), Adabas issues a BT command for the user and returns response code 9 (ADARSP009) for the OP command. If an OP command is issued by any other type of *active* user, Adabas issues a CL command for the user before processing the OP command.

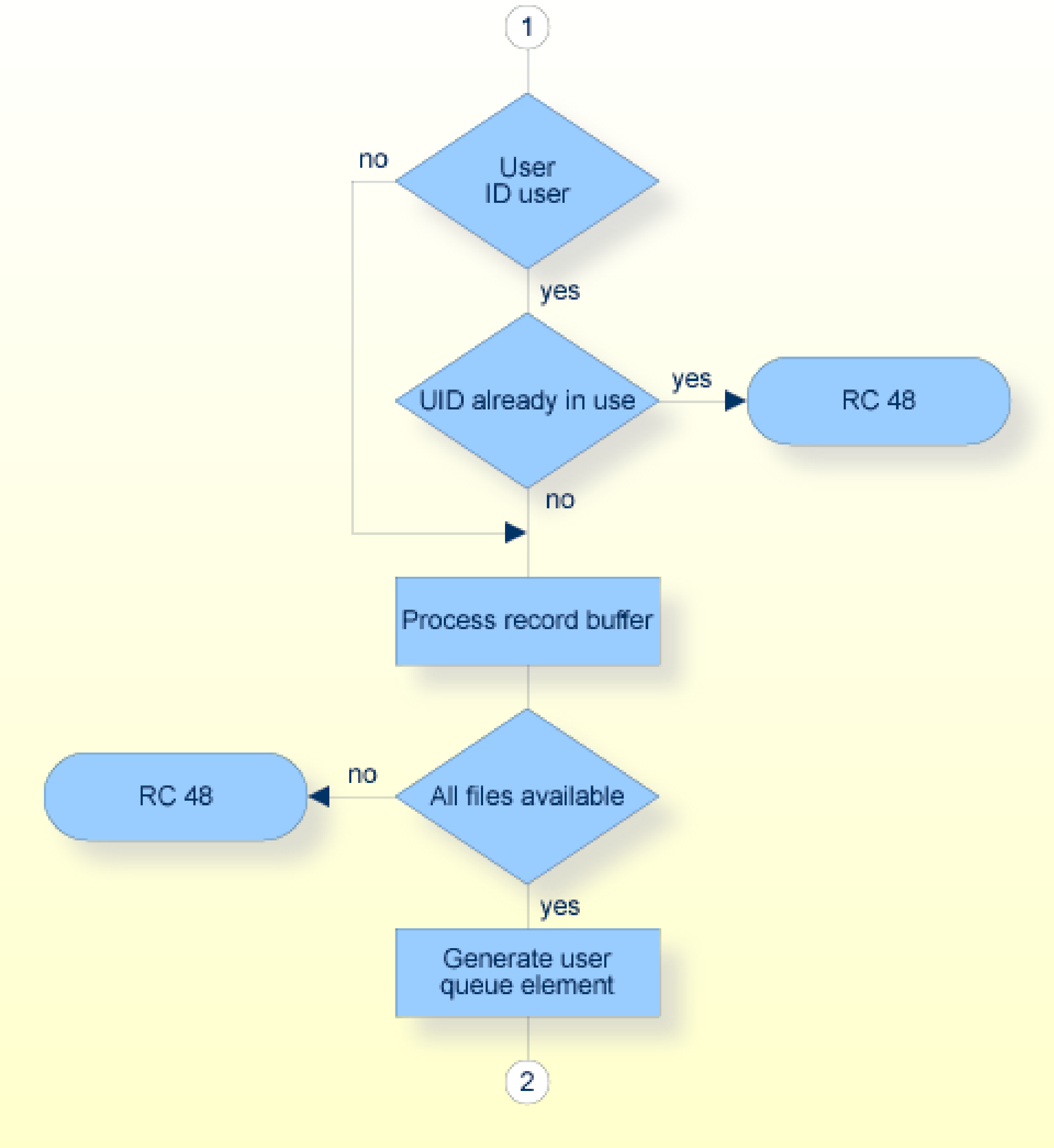
A user operating in single-user mode cannot issue more than one OP command during a given session execution.

User Types

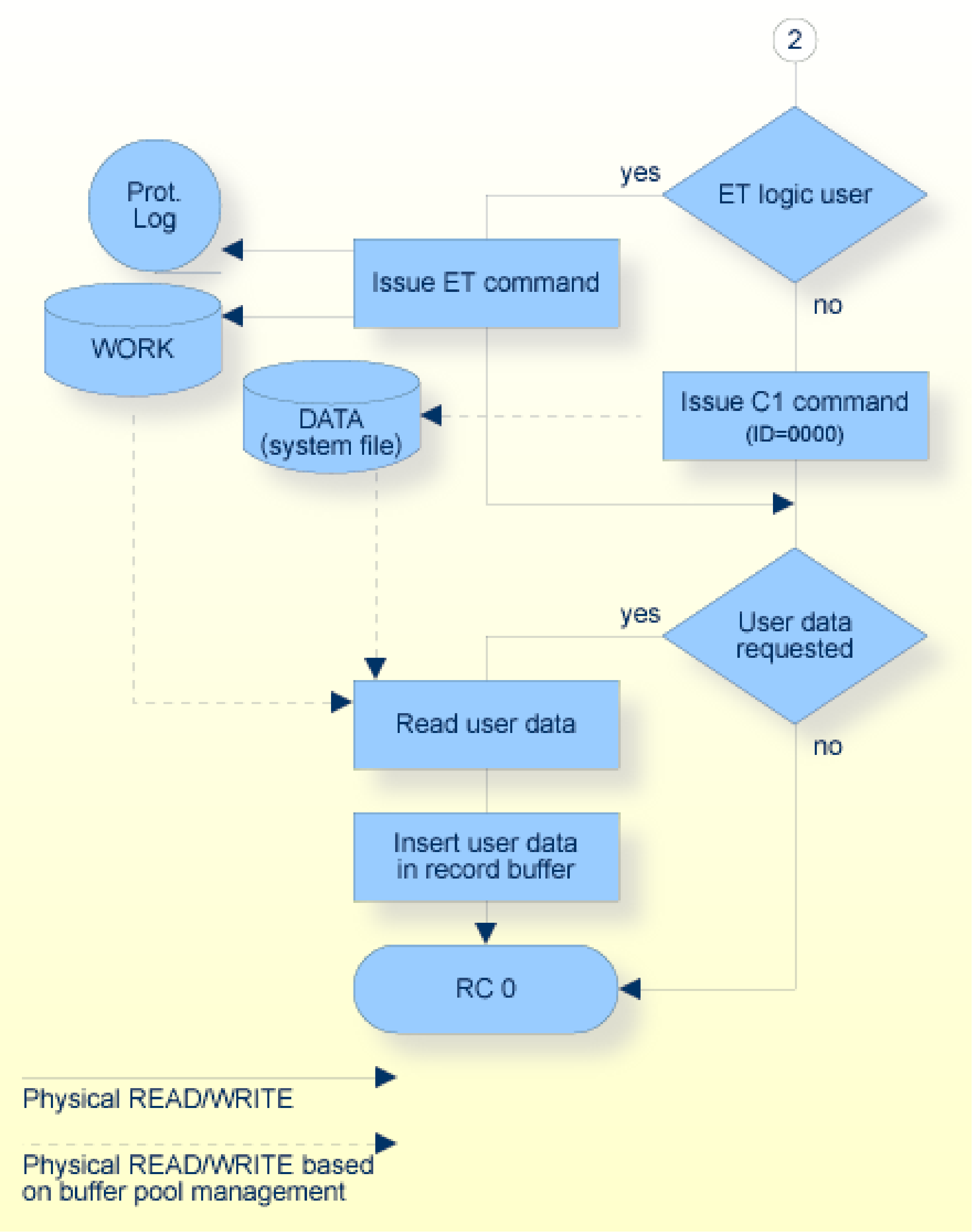
Adabas recognizes various user types depending on whether the file is updated or just accessed, and if exclusive control user was requested. This chart shows the procedure flow through the OP command.



OP Command, Procedure Flow



OP Command, Procedure Flow (continued)



OP Command, Procedure Flow (continued)

- Access-Only Users
- Exclusive Control Users

- ET Logic Users

Access-Only Users

If an OP command is issued in which access-only (ACC parameter) is specified, the user is defined to be an access-only user. Such users may not issue hold, update, delete, add record, ET, or BT commands.

Note: Access-/update-level security can also be controlled through Adabas Security on a file and field level, and through Adabas SAF Security on a database and file level. The security control supplements the user type control; that is, an access-only user's access level can be further defined on a file, field, or value level with Adabas Security, but cannot be changed to an update level.

Exclusive Control Users

If an OP command is issued in which exclusive file control (EXF or EXU parameter) is specified, the user is defined to be an exclusive control user. Such a user is considered to be a non-ET logic user (unless the UPD parameter has also been specified in which case the user is defined to be an ET logic user). If such a user issues an ET command, the user is changed to an ET logic user when the first ET command is issued.

ET Logic Users

All other users (including users who do not issue an OP command) are defined as ET logic users. Transactions issued by such users are subject to the transaction duration time limit.

ACB Interface Direct Call: OP Command

This section describes ACB interface direct calls for the OP command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions

- ACB Examples

Control Block and Buffer Information

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	--	A
File Number**	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	F	A
ISN Quantity	21-24	binary	F	A
	25-26	--	--	--
Record Buffer Length	27-28	binary	F	U
	29-34	--	--	--
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
Additions 2	45-48	alphanumeric	--	A
	49-56	--	--	--
Additions 4	57-64	binary	F	A
Additions 5	65-72	binary	--	A
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of the call statement
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

OP

Command ID (ACBCID)

Adabas will return binary zeros in this field if the previous session for this user was terminated successfully with a CL command, or no previous session existed for this user.

If the previous session for this user was not terminated successfully with a CL command, Adabas will return in this field the transaction sequence number of the last successfully completed user transaction.

The above information is returned only if the user is an ET logic user. If the user is a non-ET logic user, this field is not modified by Adabas.

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

Adabas sets this field to binary zero on return.

ISN Lower Limit: Non-activity Time Limit (ACBISL)

This field may be used to provide a user-specific non-activity time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTNA. If this field contains binary zeros, the non-activity time limit specified by the appropriate ADARUN parameter (TNAA, TNAE, or TNAX) for the Adabas session is in effect.

Following successful OP completion, Adabas returns its system, call type, and nucleus ID (nucid) information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 4 and 5. See the section *Values Returned in Control Block Fields* for detailed information.

ISN Quantity: Transaction Time Limit (ACBISQ)

This field may be used to provide a user-specific transaction time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTT. If this field contains binary zeros, the transaction time limit specified by the ADARUN TT parameter for the Adabas session is in effect.

Following successful OP completion, Adabas returns its system release information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 6 and 7. See the section *Values Returned in Control Block Fields* for detailed information.

Record Buffer Length (ACBRBL)

The length of the record buffer must be specified in this field. The length specified must be large enough to accommodate all required record buffer entries. The record buffer length should be set to zero if an empty record buffer is to be supplied.

If user data stored in an Adabas system file is to be returned, the length specified must be large enough to permit the user data to be inserted in the record buffer; otherwise, the user data will be truncated.

Command Option 1: Restrict Files Option (ACBCOP1)

Option	Description
R (restrict files)	The user is restricted to the files specified in the file list provided in the record buffer. The OP command returns a response code 48 (ADARSP048) if a specified file is not available. Later commands issue response code 17 (ADARSP017) if the user attempts to access/update a file not contained in the file list. If the "R" option is not specified and an attempt is made to access a file not currently in the specified file list, Adabas tests to determine whether the file is currently in use by an Adabas utility and if not, the file is added to the user's list. See the section *User Queue Element* for more information.

Command Option 2: Read User Data (ACBCOP2)

Option	Description
E	user (ET) data stored in an Adabas system file by the last successful C3, CL, or ET command issued by the user (in which user data was provided) is returned in the record buffer. This option may only be used if the user has provided the same user ID for this user session and the session during which the user data was stored.

Additions 1: User ID (ACBADD1)

This field may be used to provide a user ID for the user session. To avoid later limitations, Software AG recommends that you always specify a user ID.

The value provided for the user ID should be unique for the user (that is, not used by any other user at the same time), and must begin with the value "A" through "9". If the value is not unique, a response code 48 (ADARSP048) occurs. If the other user has been inactive for 60 seconds, the nucleus schedules an internal autobackout for that user and returns response code 9 (ADARSP009) for the OP command. With another OP command, the user can take over the user ID (ETID) of the other user, who loses the user ID due to the internal backout transaction (BT) command and receives response code 9 (ADARSP009) on the next call.

A user ID *must* be provided if any of the following is true:

- The user intends to first read (if any) and then store user data, and the user wishes the data to be available during a subsequent user or Adabas session.

  A user that specifies a user ID (ETID) can store ET data that remains available in later sessions. ET data stored by a user having no user ID is available during the user's current session only. Data to be stored can be provided with the ET command or at the end of the session with a close (CL) command. ET data stored with a user ID in a previous session can be read with either the OP or RE command.
- The user is to be assigned a special processing priority (priorities are assigned with Adabas Online System or the ADADBS utility's PRIORITY function);
- The operation being started is on a multiclient file; see the *Adabas DBA Reference* documentation for more information about multiclient files. A user with a blank or missing owner ID receives response code 3 (ADARSP003) or 113 (ADARSP113) when trying to access a multiclient file.

Users for whom none of the above conditions are true may set this field to blanks.

Additions 2: Transaction Sequence Number (ACBADD2)

Adabas returns in this field the transaction sequence number of the last transaction for which an ET command with ET data was executed successfully.

If the OP command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 4: Maximum Settings (ACBADD4)

This field may be used to set the following user-specific maximum binary values:

Bytes	Usage
57-58	maximum number of ISNs that may be stored in the internal ISN element table resulting from the execution of a S*x* command. Increasing the default setting will result in less access to the Adabas Work being necessary. The maximum allowed is 1000.
59-60	maximum number of records that a user may have in hold status at the same time. The default is the value set by the ADARUN NISNHQ parameter. The maximum is the smaller of either the value set by the ADARUN NH parameter or 16,777,215. If the value is set to hex 'FFFF', the value set by the ADARUN NH parameter will be used.

Bytes	Usage
61-62	maximum number of command IDs that may be active for a user at the same time. This value cannot be greater than 1/240 X LQ (where LQ is the ADARUN sequential command table length parameter value, which has a default of 10000).
63-64	maximum amount of time permitted for the execution of an S*x* command.

If one or more of the values above are not specified, the system-wide specifications for the nucleus become the defaults. The user should consult with the DBA concerning the system defaults in effect for these values before entering any user-specific values in this field.

Values specified must be in binary. Specifying blanks or binary zero is equivalent to "no value".

Adabas sets the Additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Additions 5: Returned Time-Out Values (ACBADD5)

User-specific timeout values are returned in the left (high-order) and right (low-order) halves, respectively, of the rightmost fullword of the Additions 5 field. For more information, see the section *Values Returned in Control Block Fields*.

ACB Examples

- Example 1: Access-Only User
- Example 2: ET Logic User
- Example 3: Exclusive Control User Without ET Logic
- Example 4: Exclusive Control User with ET Logic
- Example 5: Special Encoding for Wide-Character Fields

Example 1: Access-Only User

An access-only user session is to be opened.

Control Block

Command Code	OP	
Record Buffer Length	4	or larger

Buffer Areas

Record Buffer	ACC. (or ACC= . or ACC=file-list.)

Allows all selected files to be accessed.

Example 2: ET Logic User

A user session is to be opened during which the user intends to access files 8 and 9 and update files 8 and 16. The user intends to store user data in an Adabas system file during the session. The user data stored during the previous session is to be read. The ID for the user is "USER0001".

Control Block

Command Code	OP	
Record Buffer Length	15	or larger
Command Option 2	E	user data is to be read
Additions 1	USER0001	user ID is required if user data is to be stored and/or read

Buffer Areas

Record Buffer	ACC=9,UPD=8,16.

Example 3: Exclusive Control User Without ET Logic

A user session is to be opened during which the user wishes to have exclusive control of files 10, 11, and 12. The user does not intend to use ET commands and does not intend to store or read user data in an Adabas system file.

Control Block

Command Code	OP	
Record Buffer Length	13	or larger
Command Option 2	*b* (blank)	user data is not to be stored or read
Additions 1	*bbbbbbbb* (blanks)	user ID is not required

Buffer Areas

Record Buffer	EXU=10,11,12.

Example 4: Exclusive Control User with ET Logic

A user session is to be opened during which the user wishes to have exclusive control of files 10,11, and 12. The user intends to use ET commands.

Control Block

Command Code	OP	
Record Buffer Length	26	or larger
Command Option 2	*b* (blank)	user data is not to be stored or read
Additions 1	*bbbbbbbb* (blanks)	user ID is not required

Buffer Areas

Record Buffer	EXU=10,11,12,UPD=10,11,12.

Example 5: Special Encoding for Wide-Character Fields

A user session is to be opened with shift-JIS special encoding for wide-character fields. The user intends to update file number 1.

Control Block

Command Code	OP	
Record Buffer Length	16	or larger

Buffer Areas

Record Buffer	UPD=1,WCODE=932.

ACBX Interface Direct Call: OP Command

This section describes ACBX interface direct calls for the OP command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	---	A
Database ID**	17-20	numeric	F	U
	21-28	---	---	---
ISN	29-32	binary	---	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	A
	41-44	---	---	---
ISN Quantity	45-48	binary	F	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
Additions 2	65-68	binary	---	A
	69-76	---	---	---
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	---	A
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---

Field	Position	Format	Before Adabas Call	After Adabas Call
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	*	
Record	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but should be included in Adabas call or one will be automatically generated.
**	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

OP

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

Adabas will return binary zeros in this field if the previous session for this user was terminated successfully with a CL command, or no previous session existed for this user.

If the previous session for this user was not terminated successfully with a CL command, Adabas will return in this field the transaction sequence number of the last successfully completed user transaction.

The above information is returned only if the user is an ET logic user. If the user is a non-ET logic user, this field is not modified by Adabas.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

ISN (ACBXISN)

Adabas sets this field to binary zero on return.

ISN Lower Limit: Non-activity Time Limit (ACBXISL)

This field may be used to provide a user-specific non-activity time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTNA. If this field contains binary zeros, the non-activity time limit specified by the appropriate ADARUN parameter (TNAA, TNAE, or TNAX) for the Adabas session is in effect.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

Following successful OP completion, Adabas returns its system, call type, and nucleus ID (nucid) information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 4 and 5. See the section *Values Returned in Control Block Fields* for detailed information.

ISN Quantity: Transaction Time Limit (ACBXISQ)

This field may be used to provide a user-specific transaction time limit. This limit must conform to the maximum specified by the ADARUN parameter MXTT. If this field contains binary zeros, the transaction time limit specified by the ADARUN TT parameter for the Adabas session is in effect.

The ACBXISQ field is a four-byte binary field embedded in the eight-byte ACBXISQG field, which is not yet used. Set the high-order part of the ACBXISQG field to binary zeros.

Following successful OP completion, Adabas returns its system release information in this field; the timeout information previously held here is returned in the Additions 5 field, bytes 6 and 7. See the section *Values Returned in Control Block Fields* for detailed information.

Command Option 1: Restrict Files Option (ACBXCOP1)

Option	Description
R (restrict files)	The user is restricted to the files specified in the file list provided in the record buffer. The OP command returns a response code 48 (ADARSP048) if a specified file is not available. Later commands issue response code 17 (ADARSP017) if the user attempts to access/update a file not contained in the file list. If the "R" option is not specified and an attempt is made to access a file not currently in the specified file list, Adabas tests to determine whether the file is currently in use by an Adabas utility and if not, the file is added to the user's list. See the section ***User Queue Element*** for more information.

Command Option 2: Read User Data (ACBXCOP2)

Option	Description
E	user (ET) data stored in an Adabas system file by the last successful C3, CL, or ET command issued by the user (in which user data was provided) is returned in the record buffer. This option may only be used if the user has provided the same user ID for this user session and the session during which the user data was stored.

Additions 1: User ID (ACBXADD1)

This field may be used to provide a user ID for the user session. To avoid later limitations, Software AG recommends that you always specify a user ID.

The value provided for the user ID should be unique for the user (that is, not used by any other user at the same time), and must begin with the value "A" through "9". If the value is not unique, a response code 48 (ADARSP048) occurs. If the other user has been inactive for 60 seconds, the nucleus schedules an internal autobackout for that user and returns response code 9 (ADARSP009) for the OP command. With another OP command, the user can take over the user ID (ETID) of the other user, who loses the user ID due to the internal backout transaction (BT) command and receives response code 9 (ADARSP009) on the next call.

A user ID *must* be provided if any of the following is true:

- The user intends to first read (if any) and then store user data, and the user wishes the data to be available during a subsequent user or Adabas session.

  A user that specifies a user ID (ETID) can store ET data that remains available in later sessions. ET data stored by a user having no user ID is available during the user's current session only. Data to be stored can be provided with the ET command or at the end of the session with a close (CL) command. ET data stored with a user ID in a previous session can be read with either the OP or RE command.

- The user is to be assigned a special processing priority (priorities are assigned with Adabas Online System or the ADADBS utility's PRIORITY function);
- The operation being started is on a multiclient file; see the *Adabas DBA Reference* documentation for more information about multiclient files. A user with a blank or missing owner ID

receives response code 3 (ADARSP003) or 113 (ADARSP113) when trying to access a multiclient file.

Users for whom none of the above conditions are true may set this field to blanks.

Additions 2: Transaction Sequence Number (ACBXADD2)

Adabas returns in this field the transaction sequence number of the last transaction for which an ET command with ET data was executed successfully.

Additions 4: Maximum Settings (ACBXADD4)

This field may be used to set the following user-specific maximum binary values:

Bytes	Usage
77-78	maximum number of ISNs that may be stored in the internal ISN element table resulting from the execution of a S*x* command. Increasing the default setting will result in less access to the Adabas Work being necessary. The maximum allowed is 1000.
79-80	maximum number of records that a user may have in hold status at the same time. The default is the value set by the ADARUN NISNHQ parameter. The maximum is the smaller of either the value set by the ADARUN NH parameter or 16,777,215. If the value is set to hex 'FFFF', the value set by the ADARUN NH parameter will be used.
81-82	maximum number of command IDs that may be active for a user at the same time. This value cannot be greater than 1/240 X LQ (where LQ is the ADARUN sequential command table length parameter value, which has a default of 10000).
83-84	maximum amount of time permitted for the execution of an S*x* command.

If one or more of the values above are not specified, the system-wide specifications for the nucleus become the defaults. The user should consult with the DBA concerning the system defaults in effect for these values before entering any user-specific values in this field.

Values specified must be in binary. Specifying blanks or binary zero is equivalent to "no value".

Adabas sets the Additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Returned Time-Out Values (ACBXADD5)

User-specific timeout values are returned in the left (high-order) and right (low-order) halves, respectively, of the rightmost fullword of the Additions 5 field. For more information, see the section ***Values Returned in Control Block Fields***.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to OP commands:

- Format Buffer
- Record Buffer

Format Buffer

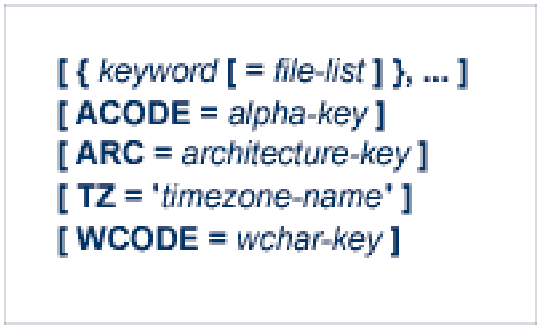
A format buffer is not used by the OP command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

The record buffer specifies:

- the files to be accessed or updated, as well as the type of updating to be performed
- special encoding for alphanumeric or wide-character fields during the session
- for fields in record and value buffers, special architecture that overrides the architecture for remote calls set by Entire Net-work.

The syntax of the record buffer is



```
[ { keyword [ = file-list ] }, ... ]
[ ACODE = alpha-key ]
[ ARC = architecture-key ]
[ TZ = 'timezone-name' ]
[ WCODE = wchar-key ]
```

where:

- `keyword` is one of the following:

ACC ESS file is to be accessed only

UPD ATE file is to be updated (implies ET logic)

EXF exclusive file control: no other users may access/update the file

EXU file is to be updated under exclusive control of the user. No other user can update the specified file while this user session is active. Exclusive control is given only if no other active user has issued an OP command specifying the EXF/EXU or UPD parameter for the file.

- *`file-list`* is one or more 1-5-digit file numbers (leading zeros permitted) indicating the Adabas file(s) for which the preceding file-list keyword is applicable.
- ACODE assigns special encoding for A-type fields during the user session.

  `alpha-key` specifies the key of supplied encoding descriptor objects.
- ARC defines a special data architecture for fields in the record and value buffers. This definition overrides the architecture key defined for remote calls in Entire Net-work.

  Note: The ARC setting does not affect the data conversion performed by ADALNK / LNKUES on the Adabas control block (ACB) and buffers with fixed layout such as the search and format buffers.

  `architecture-key` is an integer which is the sum of the following numbers:

byte order	b=0	high-order byte first
	b=1	low-order byte first
encoding family	e=0	ASCII encoding family
	e=2	EBCDIC encoding family (default for local calls)
floating-point format	f=0	IBM370 floating-point format
	f=4	VAX floating-point format
	f=8	IEEE floating-point format

  The default is ARC = b + e + f = 2; that is, high-order byte first; EBCDIC encoding family; and IBM370 floating-point format (b=0; e=2; f=0).

  User data from an Intel386 PC provides the example: b=1; e=0; f=8; or ARC=9.
- TZ assigns the time zone for the user session. Adabas uses this time zone setting to convert the user local date-time value (the fields with the TZ option) to or from the value stored in universal time (UTC). If no time zone is specified for a user session, Adabas issues respone 55 subcode 30 when attempting this conversion.

  `timezone-name` specifies the name of a valid Adabas-supported time zone. For information about the time zone names in the Adabas time zone library, read *Supported Time Zones*, in the *Adabas DBA Tasks Manual*.

Note: Time zone names are case-sensitive and must be specified in single quotation marks.

For example:

```
TZ='America/New_York'
```

- WCODE assigns special encoding for W-type fields during the user session.

  wchar-key specifies the key of supplied encoding descriptor objects.

The record buffer syntax must end with a period (".").

If no parameters are specified for the record buffer, it contains only a period ("."). In this case, the record buffer length can be set to zero in the Adabas control block and the record buffer need not be supplied.

If a user-type keyword is to apply to a series of files, each file for which the keyword is applicable may be specified with a comma between each file number. Duplicate file numbers within a keyword are permitted. Duplicate file numbers across keywords are permitted. Each keyword may appear only once.

UPD and EXU also imply access to the file. If ACC is the only keyword specified, a file list is not required.

If no user-type keyword is specified, the user automatically becomes an ET logic user.

User Queue Element

During the time that a user is active, Adabas maintains a user queue element (UQE) for the user.

User Type and File Lists

The UQE lists the numbers of up to 5000 files the user is currently using. For a non-ET user, Adabas creates the file list when the user issues an OP command; no file list is created for an ET user. The file list may be modified during the user session. If no OP command is issued, the file list will initially contain no files. Each file in the file list is marked with one of the following use classes:

- ACC, access only;
- EXF/EXU, access and update and under exclusive control;
- UPD, access and update;

- UTI, access and update and in use by an Adabas utility.

If a subsequent attempt is made to access a file not currently in the specified file list, Adabas tests to determine whether the file is currently in use by an Adabas utility. If not, the file is added to the user's UQE and marked as ACC. However, if the OP command specifies the restrict files option in the Command Option 1 field with a file list in the record buffer, and then tries to access a file *not* in the file list, response code 17 (ADARSP017) occurs.

If a subsequent attempt is made to update a file not currently in the user's file list, the following tests are applied:

- Does the request conflict with the user type? For example, an access-only user may not issue update commands.
- Is the file to be updated under exclusive control of another user or Adabas utility?

If the file is determined to be available for the user, the file is added to the user's UQE and marked as UPD.

Special Encoding Information

If the user specifies ACODE, WCODE, and/or ARC to determine the special encoding to be used, this information is communicated to the Adabas nucleus, which stores it in the UQE.

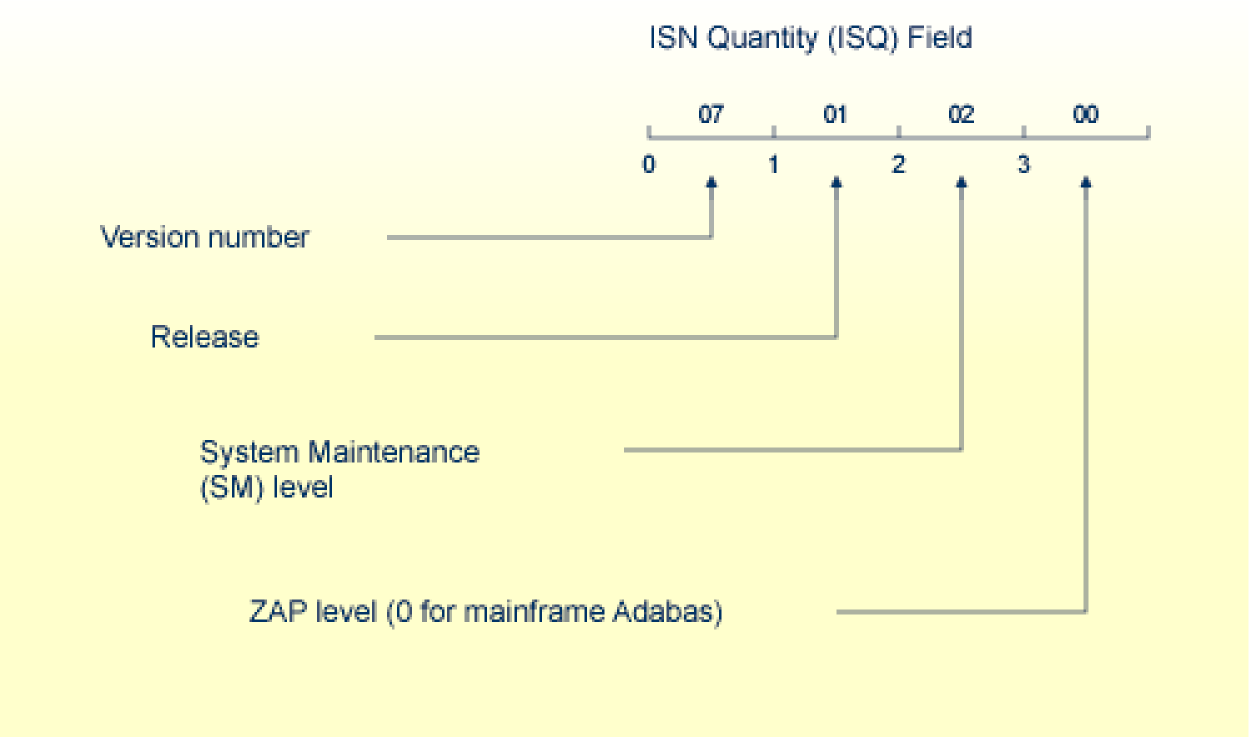
Exceeding Time Limits

See the section on timeout characteristics in the *Adabas Operations* documentation for information on which action will be taken if a user exceeds the non-activity time limit.

Values Returned in Control Block Fields

In some cases, values are returned in the ISN lower limit and ISN quantity fields to allow compatibility with VMS/UNIX systems. As a result, any user-specific timeout values previously held in these fields in early Adabas releases are now returned in the Additions 5 field. These changes also affect the corresponding command log fields.

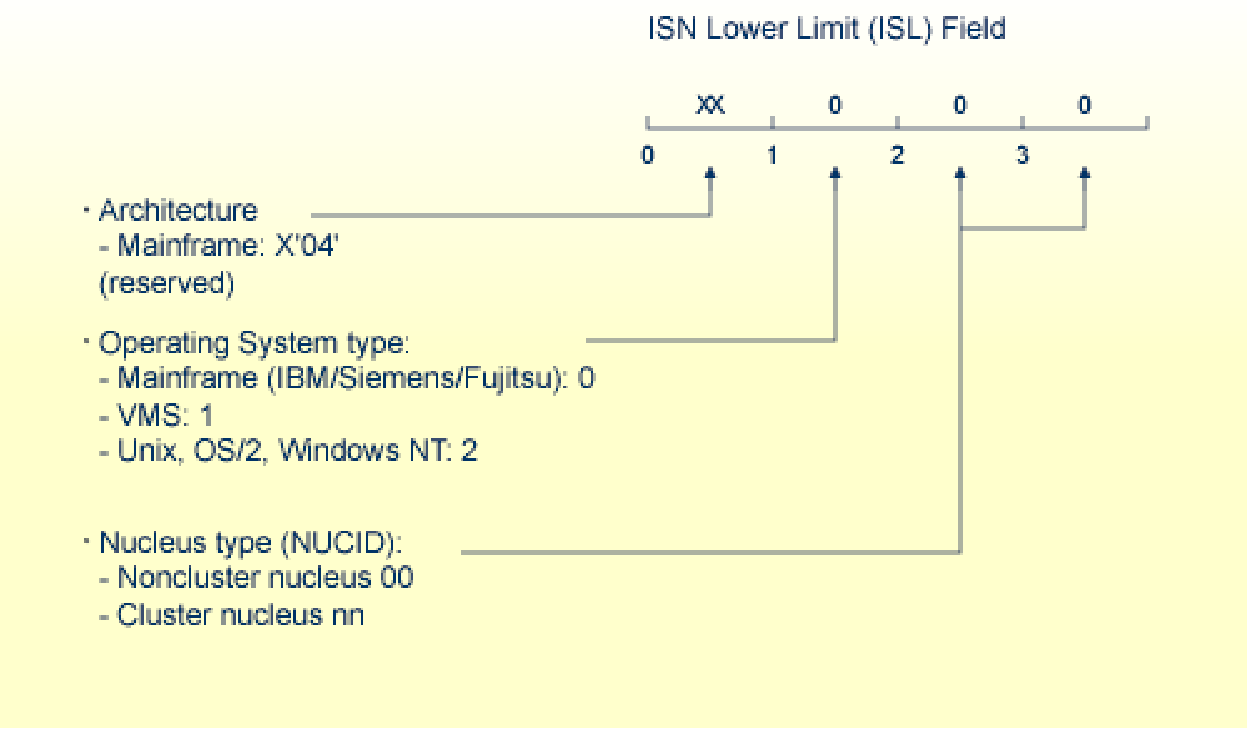
The ISN quantity field returns the following binary values after OP execution:



Binary Values Returned after OP Execution

This returned information provides compatibility with VMS/UNIX.

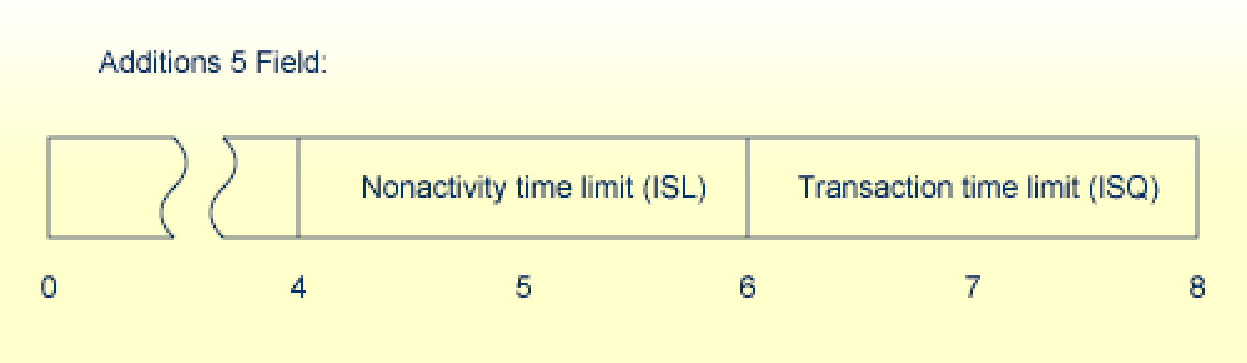
The ISN lower limit (ISL) field now returns the following binary information:



Binary Information Returned from ISL

The rightmost two bytes indicate whether the user program is running in a noncluster (00) or cluster (nn) nucleus environment. The specific cluster nucleus type return code is for use by Software AG technical support.

Any timeout values specified in the ISL/ISQ fields at the start of OP processing are returned in the Additions 5 field, as follows:



Timeout Values

and correspondingly, in the Additions 5 command log entry.

38 RC Command: Release Command ID or Global Format ID

The RC command releases one or more command IDs or a global format ID for the issuing user.

We recommend that you set unused ACB and ACBX fields to binary zeros before the direct call is initiated.

Function and Use

The RC command may be used to release one or more command IDs currently assigned to a user, or to delete one or all global format IDs, as follows:

- Internal format buffer pool command IDs. Related internal formats are also released;
- ISN list (TBI) command IDs;
- Command IDs in the table of sequential commands (TBLES/TBQ);
- Command IDs equal to and greater than the specified command ID value in either the internal format buffer pool or the TBI, TBLES and TBQ, or both;
- One special global format ID for a user group;
- All existing global format IDs.

If no selective options are specified, the entered command ID is released from all of the above areas. When a command ID is released, its related TBI or TBLES/TBQ entries are also removed; however, the internal format buffer pool entry is not necessarily released.

The RC command should be used under the following conditions:

- The user has completed processing an ISN list stored on the Adabas Work by an S*x* command that specified the save-ISN-list option. Issuing the RC command permits Adabas to reuse the space currently occupied by the list;
- The user wishes to terminate a sequential pass of a file (using an L2/L5, L3/L6, or L9 command) before reaching an end-of-file condition;
- The user has completed a series of L1/L4, A1/A4, or N1/N2 commands in which a non-blank command ID was used.

ACB Interface Direct Call: RC Command

This section describes ACB interface direct calls for the RC command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions
- ACB Examples

Control Block and Buffer Information

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
	13-34	--	--	--
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
	45-64	--	--	--
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

RC

Command ID (ACBCID)

The command ID to be released or to be used as a reference is specified in this field. A value of blanks or binary zeros releases all the command IDs currently assigned to the user.

File Number (ACBFNR)

If the Command Option 1 field is set to D, E, or O, the file number field must contain the binary number of the file associated with the format or global format ID to be released.

For physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B'0000 0000').
- For a two-byte file number, use both bytes (9 and 10) of the field.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

Command Option 1/2: Type of Command IDs to Be Released (ACBCOP1 and ACBCOP2)

These fields are used to indicate that a command ID, format ID, or global format ID is to be released from the internal format buffer pool, the ISN list table (TBI), or the table of sequential commands (TBLES/TBQ). For information about the tables, see the section *General Programming Considerations*.

If both Command Option 1/2 fields are set to blanks or binary zeros, the command ID specified in the command ID field is released from all tables in which it is present.

If either Command Option field is set to one of the following values, the resources associated with the command ID, format ID, or global format ID are released as indicated:

Option	Releases . . .
C	all existing global formats
D	all formats for a given file number and descriptor name
E	all global formats for a given file number and descriptor name
F	the format associated with the specified command ID
G	all existing formats associated with command IDs greater than or equal to the specified command ID
I	the ISN list (TBI) associated with the specified command ID
L	the global format ID contained in the Additions 5 field.
O	the global format ID contained in the Additions 5 field for a given file number.
S	the sequential commands (TBLES/TBQ) associated with the specified command ID
X	all ISN lists (TBI) and sequential commands (TBLES/TBQ) associated with command IDs that are greater than or equal to the specified command ID. Internal formats are not released.

Options D and E are used when the format was created by an L3 or L6 command to ensure the return of correct data in an environment where Smith/Jones problems are possible. The underlying format identifier in these cases is 12 bytes: an 8-byte format ID, a 2-byte file number, and a 2-byte descriptor name.

Additions 1: Descriptor Name (ACBADD1)

If Command Option D or E is specified, the first two bytes of the Additions 1 field must contain the alphanumeric descriptor field name associated with the format or global format ID to be released. All remaining positions must be set to blanks.

If the format to be released was not created using the L3 or L6 command, this field is not used.

Additions 5: Released Global Format ID (ACBADD5)

In this field, specify a global format ID to be released.

ACB Examples

- Example 1
- Example 2
- Example 3
- Example 4

- Example 5

Example 1

The command ID "0003" is to be released.

Control Block

Command Code	RC	
Command ID	X'0003'	command ID 003 to be released
Command Option 1/2	*bb* (blanks)	all CID types to be released

Example 2

All command IDs currently assigned to the user are to be released.

Control Block

Command Code	RC	
Command ID	X'00000000'	binary zeros indicate that all command IDs are to be released
Command Option 1/2	*bb* (blanks)	all CID types to be released

Example 3

All the command IDs assigned to the user and contained in the table of sequential commands or the internal format buffer pool are to be released.

Control Block

Command Code	RC	
Command ID	X'00000000'	binary zeros indicate that all command IDs are to be released
Command Option 1	F	F indicates that command IDs contained in the internal format buffer pool are to be released
Command Option 2	S	S indicates that command IDs contained in the table of sequential commands are to be released

Example 4

The same global format ID is defined for several files. Release it for all files.

Control Block

Command Code	RC	
Command Option 1/2	L	releases the formats of the global format ID contained in the Additions 5 field.
Additions 5	C'TGLOB001'	B'11' in the two high-order (leftmost) bits of the first byte of this number identify all eight bytes as the global format ID.

Example 5

The same global format ID is defined for several files. Release it for the file 3 only.

Control Block

Command Code	RC	
File Number	03	binary number of the file for which the global format ID is to be released.
Command Option 1/2	O	releases the formats of the global format ID contained in the Additions 5 field for the file specified in the file number field.
Additions 5	C'TGLOB001'	B'11' in the two high-order (leftmost) bits of the first byte of this number identify all eight bytes as the global format ID.

ACBX Interface Direct Call: RC Command

This section describes ACBX interface direct calls for the RC command. It covers the following topics:

- Control Block and Buffer Information

- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-48	---	---	---
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
	65-84		---	---
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

RC

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

The command ID to be released or to be used as a reference is specified in this field. A value of blanks or binary zeros releases all the command IDs currently assigned to the user.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

If the Command Option 1 field is set to D, E, or O, use this field to specify the number of the file associated with the format or global format ID to be released.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

Command Options 1 and 2: Type of Command IDs to Be Released (ACBXCOP1 and ACBXCOP2)

These fields are used to indicate that a command ID, format ID, or global format ID is to be released from the internal format buffer pool, the ISN list table (TBI), or the table of sequential commands (TBLES/TBQ). For information about the tables, see the section ***General Programming Considerations***.

If both Command Option 1/2 fields are set to blanks or binary zeros, the command ID specified in the command ID field is released from all tables in which it is present.

If either Command Option field is set to one of the following values, the resources associated with the command ID, format ID, or global format ID are released as indicated:

Option	Releases . . .
C	all existing global formats
D	all formats for a given file number and descriptor name
E	all global formats for a given file number and descriptor name
F	the format associated with the specified command ID
G	all existing formats associated with command IDs greater than or equal to the specified command ID
I	the ISN list (TBI) associated with the specified command ID
L	the global format ID contained in the Additions 5 field.
O	the global format ID contained in the Additions 5 field for a given file number.
S	the sequential commands (TBLES/TBQ) associated with the specified command ID
X	all ISN lists (TBI) and sequential commands (TBLES/TBQ) associated with command IDs that are greater than or equal to the specified command ID. Internal formats are not released.

Options D and E are used when the format was created by an L3 or L6 command to ensure the return of correct data in an environment where Smith/Jones problems are possible. The underlying format identifier in these cases is 12 bytes: an 8-byte format ID, a 2-byte file number, and a 2-byte descriptor name.

Additions 1: Descriptor Name (ACBXADD1)

If Command Option D or E is specified, the first two bytes of the Additions 1 field must contain the alphanumeric descriptor field name associated with the format or global format ID to be released. All remaining positions must be set to blanks.

If the format to be released was not created using the L3 or L6 command, this field is not used.

Additions 5: Released Global Format ID (ACBXADD5)

In this field, specify a global format ID to be released.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

39 RE Command: Read ET User Data

The RE command reads ET (user) data for the current user, another user, or all users.

Function and Use

The RE command reads user data that was previously stored in an Adabas system checkpoint file by a C3, CL, or ET command. The user data is returned in the **record buffer**. This user data may be needed for a user restart following abnormal termination of a user or Adabas session.

User data is read for the issuing user if no value is specified in the Command Option 1 field. The RE command reads user data stored during a previous session for the issuing user if the previous and current sessions both began with OP commands specifying the user ID.

Depending on the specified Command Option, the RE command reads user data for either another specific user (if that user ID is specified) or for all users.

- If "I" is specified in the Command Option 1 field, user data stored by another user may be read if the ID of the user who stored the data is specified in the Additions 1 field.
- If "A" is specified in the Command Option 1 field, the current and the following RE commands read all user data for all user IDs in ascending logical sequence. The corresponding user ID is returned in the Additions 1 field as each RE command is completed. The RE command with option "A" reads only the user data written to the checkpoint file whose transactions ended with an ET command; user data is *not* read for users' transactions that are not yet closed with an ET command.

ACB Interface Direct Call: RE Command

This section describes ACB interface direct calls for the RE command. It covers the following topics:

- ACB Examples

Control Block and Buffer Information

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	binary	--	A
File Number ****	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	A
	17-26	--	--	--
Record Buffer Length	27-28	binary	F	U
	29-34	--	--	--
Command Option 1	35	alphanumeric	F	U
	36	--	--	--
Additions 1	37-44	alphanumeric	F *	U/A **
Additions 2	45-48	alphanumeric	--	A
	49-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	***	--
Record	--	A

where:

F Supplied by user before Adabas call

A Supplied by Adabas

U Unchanged after Adabas call

* Supplied ET data user ID when Command Option 1 equals "I"

** User ID for ET data in record buffer if Command Option 1 equals "A"

*** Not used but must be included in parameter list of call statement

**** A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

-- Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

RE

Command ID (ACBCID)

Adabas will return a transaction sequence number or binary zeros in this field.

Adabas returns binary zeros in this field if the user whose user data is to be read is not active *and* either no previous session exists for this user or the previous session for this user was terminated successfully with a CL command. Non-ET-logic users receive binary zeros in this field.

If the user is currently active *or* the previous session for the user was not terminated successfully with a CL command, Adabas returns the transaction sequence number of the last successfully completed user transaction.

File Number (ACBFNR)

A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. If Command Option 1 specified "A", a response code 3 (ADARSP003) indicates end-of-file for the user data. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

If Command Option 1 specifies "A", user data is returned in logical sequence starting with this ISN. If this field specifies zero, all user data is returned.

Record Buffer Length (ACBRBL)

The length of the record buffer. The length specified determines the number of bytes of user data to be returned.

If the length specified is less than the number of bytes of user data available, only the specified number of bytes are inserted in the record buffer and the rightmost bytes are truncated.

Command Option 1 (ACBCOP1)

If no value is specified in the Command Option 1 field, user data is read for the issuing user. The RE command reads user data stored during a previous session for the issuing user if the previous and current sessions both began with OP commands specifying the user ID.

If a Command Option 1 value is specified, the RE command reads user data for either another specific user or for all users as follows:

Option	Description
I (ID of user)	Reads user data stored by another user if the ID of the user who stored the data is specified in the Additions 1 field.
A (all users)	The current and following RE commands read all user data in the record buffer for all user IDs in ascending logical (ISN) sequence. A starting ISN can be specified in the ISN field. If the ISN field specifies zero, all user data is returned. The user ID for the user data contained in the record buffer at the end of the current and each following RE operation is returned in the Additions 1 field as each RE command is completed. RE reads only the user data written to the checkpoint file for users whose transactions ended with an ET command; user data is *not* read for users' transactions that are not yet closed with an ET command. For this option, a response code 3 (ADARSP003) returned in the response code field indicates end-of-file for the user data.

Additions 1: User ID (ACBADD1)

If user data stored by another user is to be read, this field must be set to the ID of the user who stored the data. If Command Option 1 specifies "A", this field returns the user ID for the user data contained in the record buffer at the end of this and each following RE operation.

Additions 2: Transaction Sequence Number (ACBADD2)

If an ET logic user stored the data being read, Adabas will return, in this field, the transaction sequence number of the user's last successfully completed transaction in which user data was stored with an ET or CL command.

If the RE command returns a non-zero response code, the rightmost two bytes of the Additions 2 field may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

ACB Examples

- Example 1
- Example 2
- Example 3

Example 1

The user wishes to read the user's own data previously stored with an ET command.

Control Block

Command Code	RE	
Record Buffer Length	100	100 bytes of user data is to be read
Command Option 1	blank	the user data to be read was stored by this user
ISN	0	

Example 2

The user wishes to read user data stored by another user (user ID = USER0002).

Control Block

Command Code	RE	
Record Buffer Length	150	150 bytes of user data is to be read
Command Option 1	I	the user data to be read was stored by another user
Additions 1	USER0002	ID of the user who stored the user data
ISN	0	

Example 3

In the following example, the user wishes to read all user data and the corresponding user ID.

Control Block

Command Code	RE	
Record Buffer Length	250	250 bytes of user data is to be read, per user
Command Option 1	A	
ISN	0	read all user data

ACBX Interface Direct Call: RE Command

This section describes ACBX interface direct calls for the RE command. It covers the following topics:

Control Block and Buffer Information

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	---	A
Database ID****	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	A
	33-48	---	---	---
Command Option 1	49	alphanumeric	F	U
	50-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F*	U/A**
Additions 2	65-68	binary	---	A
	69-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	***	--
Record	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Supplied ET data user ID when Command Option 1 equals "I"
**	User ID for ET data in record buffer if Command Option 1 equals "A"
***	Not used but should be included in Adabas call or one will be automatically generated.
****	A database ID is only necessary if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine).
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

RE

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

If Command Option 1 specified "A", a response code 3 (ADARSP003) indicates end-of-file for the user data.

Command ID (ACBXCID)

Adabas will return a transaction sequence number or binary zeros in this field.

Adabas returns binary zeros in this field if the user whose user data is to be read is not active *and* either no previous session exists for this user or the previous session for this user was terminated successfully with a CL command. Non-ET-logic users receive binary zeros in this field.

If the user is currently active *or* the previous session for the user was not terminated successfully with a CL command, Adabas returns the transaction sequence number of the last successfully completed user transaction.

Database ID (ACBXDBID)

Use this field to specify the database ID only if you are accessing a database other than the application's default database (read in by ADARUN DBID parameter, provided in the loaded link globals table, or linked with the link routine). The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file to which the Adabas call should be directed.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

If Command Option 1 specifies "A", user data is returned in logical sequence starting with this ISN. If this field specifies zero, all user data is returned.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

Command Option 1 (ACBXCOP1)

If no value is specified in the Command Option 1 field, user data is read for the issuing user. The RE command reads user data stored during a previous session for the issuing user if the previous and current sessions both began with OP commands specifying the user ID.

If a Command Option 1 value is specified, the RE command reads user data for either another specific user or for all users as follows:

Option	Description
I (ID of user)	Reads user data stored by another user if the ID of the user who stored the data is specified in the Additions 1 field.
A (all users)	The current and following RE commands read all user data in the record buffer for all user IDs in ascending logical (ISN) sequence. A starting ISN can be specified in the ISN field. If the ISN field specifies zero, all user data is returned. The user ID for the user data contained in the record buffer at the end of the current and each following RE operation is returned in the Additions 1 field as each RE command is completed. RE reads only the user data written to the checkpoint file for users whose

Option	Description
	transactions ended with an ET command; user data is *not* read for users' transactions that are not yet closed with an ET command. For this option, a response code 3 (ADARSP003) returned in the response code field indicates end-of-file for the user data.

Additions 1: User ID (ACBXADD1)

If Command Option 1 specifies an "I", this field must be set to the user ID of the user who stored the data.

If Command Option 1 specifies "A", this field returns the user ID for the user data contained in the record buffer at the end of this and each following RE operation.

Additions 2: Transaction Sequence Number (ACBXADD2)

If an ET logic user stored the data being read, Adabas will return, in this field, the transaction sequence number of the user's last successfully completed transaction in which user data was stored with an ET or CL command.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to the RE command:

- Format Buffer
- Record Buffer

Format Buffer

A format buffer is not used by the RE command, but should be included in the Adabas call. If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call** and a format buffer is not specified, one will be automatically generated.

Record Buffer

Adabas returns the user data in the record buffer. If no user data was found, this contains blanks at the end of RE operation.

40 RI Command: Release Record

The RI command releases a held record and ISN, although records are not released unconditionally, as described in **Function and Use**, next in this chapter.

Function and Use

The RI command releases ISNs for records being held by the issuing user. The selected ISN for a single database file, or all ISNs held by the issuing user in all files can be released.

Effective with Adabas 8.2.2, records are *no longer* released unconditionally. If your application issues an RI command for a record that has been updated in the current transaction, Adabas will now return response code 113 (ADARSP113), or if ISN=0 was specified, response code 2 (ADARSP002). This behavior may affect how your application programs are coded. If your application programs perform updates and if they try to release updated records from hold status, they may be affected. Such application programs should be adjusted to either not issue RI commands for updated records or suppress the resulting response code 113 (ADARSP113). (The ADARUN RIAFTERUPDATE parameter can be used to suppress the response code 113 and response code 2 results, and only records that have not been updated in the current transaction will be released from hold.) In addition, if your installation has Natural installed, you must apply Natural zap NA76045 for downward compatibility, which is available using **Empower**.

To use this command, specify the file and ISN of the record to be released in the appropriate Adabas control block fields. Specifying zeros in the ISN field releases all the records currently being held by the user in all files.

Note: Programs using ET logic should not release records with the RI command if any updating has been performed during the current transaction, since this could result in a loss of data integrity. ET users should release ISNs with the ET or CL commands.

ACB Interface Direct Call: RI Command

This section describes ACB interface direct calls for the RI command. It covers the following topics:

Control Block and Buffer Information

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
	5-8	--	--	--
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	U
	17-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

RI

File Number (ACBFNR)

The number of the file containing the record to be released.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

The ISN of the record to be released. If an ISN value is entered, the file number must also be specified. To release all ISNs held by the user, set this field to binary zeros.

ACB Examples

- Example 1
- Example 2

Example 1

The record identified by ISN 3 in file 2 is to be released from hold status.

Control Block

Command Code	RI	
File Number	2	record to be released is in file 2
ISN	3	record with ISN 3 is to be released

Example 2

Any records being held by the issuing user are to be released from hold status.

Control Block

Command Code	RI	
File Number	-	a value in this field is ignored if the ISN field is zero
ISN	0	release ISNs for all held records from all files

ACBX Interface Direct Call: RI Command

This section describes ACBX interface direct calls for the RI command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
	13-16	---	---	---
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	U
	33-50	---	---	---
Command Option 3	51	alphanumeric	F	U
	52-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

None used.

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
RI

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)
Use this field to specify the number of the file containing the record to be released.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)
The ISN of the record to be released. If an ISN value is entered, the file number must also be specified. To release all ISNs held by the user, set this field to binary zeros.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

Command Option 3: Shared Hold Status (ACBXCOP3)

Option	Description
S	Puts the record in shared hold status until the end of the transaction. When specified in an RI request, if the record is in exclusive hold status and has not been updated in the current transaction, it is placed in shared hold status; if it is in exclusive hold status and has been updated, it is not placed in shared hold status and the exclusive hold remains in effect.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

41 S1, S2, and S4 Commands: Find Records

The S1, S2, and S4 commands return a count of records and ISNs of records satisfying the search criterion.

Function and Use

The S1, S2, and S4 commands are used to select records that satisfy a given search criterion. They can be performed on Adabas expanded files as well.

The result of an S1, S2, or S4 command is the number of records which satisfy the query and a list of the records' ISNs. The S1/S4 commands return the ISNs, sorted in *ascending* sequence; the S2 command returns the ISNs according to a sort sequence specified in the Additions 1 field. Regardless of sequence, the ISNs are returned to the **ISN buffer**.

Note: ISN lists that are not in ascending sequence cannot be processed by a later S8 command.

The following types of searches are possible:

- Single file search. The search criterion consists of one or more fields contained in a single file.
- Multiple file search using physically coupled files. The search criterion consists of fields contained in two or more files that have been physically coupled using the ADAINV utility.
- Search using the soft coupling feature. This feature provides for a combination of search, read, and internal list matching.

A search criterion may contain one or more fields that are not defined as descriptors. If nondescriptors are used, Adabas performs read operations to determine which records to return to the user. If only descriptors are used within the search criterion, Adabas resolves the query by using the Associator inverted lists; no read operation is required.

Note: The behavior of nondescriptor searches in Adabas databases differs between mainframe and open systems in regards to null suppression in the fields. In open systems, nondescriptor searches do not return records with null values in a field if the field is null-suppressed (NU); on mainframe systems, the null-suppression (NU) of fields is ignored during nondescriptor searches. At this time, to resolve this problem, we recommend that you remove the null suppression option (NU) for open systems fields, if the fields must be used for a nondescriptor search.

If a valid command ID is specified, Adabas will store on Adabas Work any ISNs that could not be inserted in the ISN buffer on the initial S1, S2, or S4 command. These overflow ISNs may be retrieved to the ISN buffer later with additional S1, S2, or S4 commands that specify the same command ID.

Adabas releases an overflow ISN list when the last ISN in the list has been returned to the user. If the user needs to retain the entire ISN list indefinitely, the save-ISN-list option may be used. If this option is specified, the entire ISN list is stored on Adabas Work. The ISN list is not released until either an RC, CL, or S*x* command with the release CID option is issued, or the Adabas session is terminated.

If the user intends to use the GET NEXT option of an L1 or L4 command to read the records identified by the ISNs, neither the ISN buffer entry nor the save-ISN-list option is required. In this case, the L1 and L4 commands obtain the ISNs automatically from the ISN list stored by Adabas.

By placing field names in the format buffer, the user can read the field's contents from the record of the first ISN in the resulting ISN list. The field's contents are read into the **record buffer**. The S4 command also places the first ISN of the resulting ISN list in hold status.

ACB Interface Direct Call: S1, S2, and S4 Commands

This section describes ACB interface direct calls for the S1, S2, and S4 commands. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions
- ACB Examples

Control Block and Buffer Information

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	F	U
ISN Quantity	21-24	binary	F *	A
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
Search Buffer Length	29-30	binary	F	U
Value Buffer Length	31-32	binary	F	U
ISN Buffer Length	33-34	binary	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
Additions 2	45-48	binary / binary	--	A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
Additions 5	65-72	alphanumeric	F	U
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	--	A
Search	F	U
Value	F	U
ISN	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Optional timeout value, in seconds
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

S1, S2, or S4

Command ID (ACBCID)

This value identifies the resulting complete or overflow ISN list stored on Adabas Work, and identifies the format buffer for subsequent commands if the read-first-record option is in effect (see the section *Format Buffer*).

If the save-ISN-list option is to be used, or if overflow ISNs are to be stored, a non-blank, non-zero value must be provided in this field.

The first byte of this field may not be set to hexadecimal 'FF'.

See the section *ISN List Processing* for more information.

File Number (ACBFNR)

File number specifies the number of the file from which the ISNs are to be selected.

If a query using coupled files is to be performed, the file specified in this field will be considered the "primary" file. The file number of physically coupled files must be no greater than 255.

The search can also be performed on Adabas expanded files.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is set to zeros.

ISN Lower Limit (ACBISL)

Use this field in the first S1 or S4 call to specify a minimum ISN value for the resulting ISN list. The list will then contain only the ISNs greater than the ISN specified in this field. If this field is set to zeros, Adabas will return all qualifying ISNs.

Note: Note this will not work for an S2 call. For sorted ISN lists (S2 commands), this field must be set either to zero or to a valid ISN.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from Adabas Work.

ISN Quantity (ACBISQ)

This field defines the maximum number of seconds that can be used for S*x* command execution.

As a result of an initial S*x* call, Adabas returns the number of records that satisfy the search criterion in this field. If security-by-value is being used, response code 1 (ADARSP001) is returned in this field along with the value 0 (one record found) or 1 (more than one record found). For more information, see the *Adabas Security Manual*.

As a result of a subsequent S*x* call used to retrieve ISNs from Adabas Work, Adabas provides the number of returned ISNs in this field. The ISNs themselves are returned in the ISN buffer.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The format buffer area defined in the user program must be as large as (or larger than) the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The record buffer area defined in the user program must be as large as (or larger than) the length specified.

Search Buffer Length (ACBSBL)

The search buffer length (in bytes). The search buffer area defined in the user program must be as large as (or larger than) the length specified.

Value Buffer Length (ACBVBL)

The value buffer length (in bytes). The value buffer area defined in the user program must be as large as (or larger than) the length specified.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs will be inserted in the ISN buffer. This field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas will reduce the length to the next lower integer which is a multiple of 4.

Command Option 1 (ACBCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S*x* command on Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
R (return)	For an S4 command, returns response code 145 (ADARSP145) if a record to be read and held is not available.

Command Option 2 (ACBCOP2)

Option	Description
D (descending sequence)	For an S2 command, sorts descriptor values in descending sequence.

If no Command Option 2 is specified for an S2 command, the descriptor values are sorted in ascending sequence.

Command Option 1/2: Release CID Option (ACBCOP1 and ACBCOP2)

The I option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field as the first action taken during command execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Additions 1: S2 Command, Descriptors Used for Sort Control (ACBADD1)

If the S2 command is being used, this field must specify the descriptor (or descriptors) to be used to control the sort sequence; if no sort argument is specified, the S2 command returns response code 28 (ADARSP028).

One to three descriptors, including subdescriptors and superdescriptors, can be specified. Phonetic descriptors or descriptors contained within a periodic group cannot be specified. A multiple-value field can be specified, in which case the ISNs will be sorted according to the lowest value present within a given record.

Any unused positions of this field must be set to blanks. For example:

```
XXYYbbbb
```

where:

XX	is the major sort descriptor; and
YY	is the minor sort descriptor
bbbb	blanks

The number of ISNs that can be sorted depends on the size of the sort work area (ADARUN LS parameter) defined by the DBA. If the sort area is too small, no sort will be performed; response code 1 (ADARSP001) will be returned, and the ISNs will be returned in ascending sequence.

Additions 2: Length of Compressed and Decompressed Record (ACBADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

If the S*x* command returns a non-zero response code, the rightmost two bytes may contain a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code and Version/Nucleus ID (ACBADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and the database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBADD5)

This field may be used to provide a separate format ID to identify the internal format buffer to be used for this command, or to provide a global format ID.

If the high-order bit of the Additions 5 field is zero (0), the value provided in the command ID field is also used as the format ID.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

For more information, refer to the section ***Command, Format, and Global Format IDs***.

ACB Examples

For the Adabas file definitions used in all the examples in this section, see ***File Definitions Used in Examples***.

- Example 1
- Example 2
- Example 3
- Example 4
- Example 5
- Example 6
- Example 7

- Example 8

Example 1

Select the records in file 1 that contain a value in the range "A" to "J" for the descriptor AA.

Control Block

Command Code	S1	
Command ID	*bbbb* (blanks)	no ISNs are to be stored on the Adabas Work
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	1	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	200	no more than 50 ISNs are expected
Command Option 1	*b* (blank)	save-ISN-list option not used
Additions 3	*bbbbbbbb* (blanks)	the file is not security-protected

Buffer Areas

Format Buffer	.	no read to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	AJ	

Example 2

Find with read option. Select the ISN of the record containing the value "ABCDEFGH" for the field AA in file 1. Also, read the record from Data Storage and return the value for the field AC.

Control Block

Command Code	S1	
Command ID	*bbbb* (blanks)	no ISNs are to be stored on the Adabas Work
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	3	or larger
Record Buffer Length	20	or larger
Search Buffer Length	3	or larger
Value Buffer Length	8	or larger
ISN Buffer Length	4	no more than one ISN is expected

Command Option 1	*b* (blank)	save-ISN-list option not used
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	AC.	the value for field AC is to be returned
Search Buffer	AA.	
Value Buffer	ABCDEFGH	

Example 3

Find with ISN buffer overflow. Select the records that contain any value in the range "A" to "D" for field AA in file 1. Use ISN buffer overflow handling.

Control Block

Command Code	S1	
Command ID	ABCD	a non-blank command ID is required
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be returned
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	100	up to 25 ISNs will be returned with each call
Command Option 1	*b* (blank)	save-ISN-list option not used
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected

Buffer Areas

Format Buffer	.	no read to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	AD	

Adabas will return, as a result of the initial S1 call, a maximum of 25 ISNs in the ISN buffer. If more than 25 ISNs resulted from the query, the remaining ISNs will be stored on Adabas Work under the command ID "ABCD". These overflow ISNs may be retrieved by repeating the S1 call using the same command ID.

Example 4

Find with save-ISN-list option. Select all the records containing the value "+80" for the field XB in file 2. Store the entire resulting ISN list on the Adabas Work.

Control Block

Command Code	S1	
Command ID	BCDE	a non-blank command ID is required when using the save-ISN-list option
File Number	2	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	3	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	200	a maximum of 50 ISNs will be returned on each call
Command Option 1	H	save-ISN-list option is to be used
Additions 3	password	file is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	XB.	
Value Buffer	X'080C'	The "080C" notation is actually a two-byte hexadecimal representation of postitive (+) packed number 80. The `X'   '` wrapping should not be specified as part of the value buffer.

The user may retrieve any group of ISNs from the ISN list that is stored as a result of this call by repeating the S1 command using the command ID "BCDE". Adabas will insert as many ISNs as can be accommodated in the ISN buffer starting with the first ISN that is greater than the ISN specified in the ISN lower limit field.

Example 5

Find with sort. Select all records containing a value in the range "A" to "F" for the field AA in file 1. Return the resulting ISN list in ascending order of the values for field AB.

Control Block

Command Code	S2	
Command ID	CDEF	a non-blank command ID is required when using the S2 command
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	12	or larger
Value Buffer Length	2	or larger
ISN Buffer Length	100	a maximum of 25 ISNs will be returned on each call
Command Option 1	b	the save-ISN-list option is not used
Command Option 2	b	the descending sort option is not used
Additions 1	AB*bbbbbb*	resulting ISNs are to be sorted on the values of field AB, where *bbbbbb* represents blanks.
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	AA,1,S,AA,1.	
Value Buffer	AF	

Example 6

Find with hold. Select the record in file 1 containing the value "87654321" for field AA. Also, read the record and place it in hold status. Return the values for fields AB and AC.

Control Block

Command Code	S4	
Command ID	*bbbb* (blanks)	blank command ID may be used since save-ISN-list option is not to be used and no overflow ISNs are expected
File Number	1	
ISN Lower Limit	0	all qualifying ISNs are to be selected
Format Buffer Length	6	or larger
Record Buffer Length	22	or larger
Search Buffer Length	3	or larger
Value Buffer Length	8	or larger
ISN Buffer Length	4	only one ISN is expected

Command Option 1	*b* (blank)	the save-ISN-list option is not to be used
Additions 3	*bbbbbbbb* (blanks)	file is not security-protected
Additions 4	*bbbbbbbb* (blanks)	file is not ciphered

Buffer Areas

Format Buffer	AB,AC.	the record identified by the first ISN is to be read, values for fields AB and AC are to be returned
Search Buffer	AA.	
Value Buffer	87654321	

Example 7

Find using coupled files. Select the records in file 1 containing the value "+100" for the field AB that are coupled to records in file 2 containing the value 'ABCDE' for the field RB.

Control Block

Command Code	S1	
Command ID	EFGH	a non-blank command ID is used since ISN overflow may occur
File Number	1	file 1 is the primary file
ISN Lower Limit	0	select all qualifying ISNs
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	14	or larger
Value Buffer Length	12	or larger
ISN Buffer Length	100	return a maximum of 25 ISNs with each call
Command Option 1	b	save-ISN-list option is not to be used
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	/1/AB,D,/2/RB.	
Value Buffer	X'100CC1C2C3C4C54040404040'	The "100CC1C2C3C4C54040404040" notation is actually a hexadecimal representation of positive (+) packed number 100 and the letters ABCDEF, followed by five spaces. The `X'  '` wrapping should not be specified as part of the value buffer.

Because file 1 was specified as the primary file, the resulting ISNs will be from file 1. If ISNs from file 2 are also desired, the find may be repeated with file number 2 specified in the file number field. The order of the search criteria in the search buffer need not be changed.

Example 8

Find using multiple search criteria (complex search). Select the set of records in file 2 containing a value of "ABCD" for subdescriptor SA, a value less than "80" for field XB, and a value in the range "MMMMM" through "ZZZZZ" (but not "S*bbbb*" through "TZZZZ") for field XE.

Control Block

Command Code	S1	
Command ID	GGGG	a non-blank command ID is used since the save-ISN-list option is to be used
File Number	2	
ISN Lower Limit	0	select all qualifying ISNs
Format Buffer Length	1	or larger
Record Buffer Length	0	or larger
Search Buffer Length	35	or larger
Value Buffer Length	27	or larger
ISN Buffer Length	0	no ISNs to be returned in the ISN buffer
Command Option 1	H	save-ISN-list option is to be used
Additions 3	password	file 2 is security-protected

Buffer Areas

Format Buffer	.	no read is to be done
Search Buffer	SA,D,XB,3,U,LT,D,XE,S,XE,N,XE,S,XE.	
Value Buffer	ABCD080MMMMMZZZZZS*bbbb*TZZZZ	

ACBX Interface Direct Call: S1, S2, and S4 Commands

This section describes ACBX interface direct calls for the S1, S2, and S4 commands. It covers the following topics:

- Control Block and Buffer Information

- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	---	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-44	---	---	---
ISN Quantity	45-48	binary	F*	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
Command Option 3 (S4 only)	51	alphanumeric	F	U
	52-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
Additions 5	85-92	alphanumeric/ binary	F	U
	93-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-128	---	---	---
Compressed Record Length	129-136	binary	---	A
Decompressed Record Length	137-144	binary	---	A
Command Time	145-152	binary	---	A

Field	Position	Format	Before Adabas Call	After Adabas Call
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	---	A
Search	F	U
Value	F	U
ISN	---	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Optional timeout value, in seconds
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

S1, S2, or S4

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

This value identifies the resulting complete or overflow ISN list stored on Adabas Work, and identifies the format buffer for subsequent commands if the read-first-record option is in effect (see the section ***Format Buffer***).

If the save-ISN-list option is to be used, or if overflow ISNs are to be stored, a non-blank, non-zero value must be provided in this field.

The first byte of this field may not be set to hexadecimal 'FF'.

See the section *ISN List Processing* for more information.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file from which the ISNs are to be selected.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

Note: For commands that operate on a coupled file pair, this field specifies the primary file from which ISNs or data are returned. The file number of physically coupled files must be no greater than 255.

The search can also be performed on Adabas expanded files.

ISN (ACBXISN)

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is set to zeros.

ISN Lower Limit (ACBXISL)

Use this field in the first S1 or S4 call to specify a minimum ISN value for the resulting ISN list. The list will then contain only the ISNs greater than the ISN specified in this field. If this field is set to zeros, Adabas will return all qualifying ISNs.

Note: Note this will not work for an S2 call. For sorted ISN lists (S2 commands), this field must be set either to zero or to a valid ISN.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from Adabas Work.

ISN Quantity (ACBXISQ)

This field defines the maximum number of seconds that can be used for S*x* command execution.

The ACBXISQ field is a four-byte binary field embedded in the eight-byte ACBXISQG field, which is not yet used. Set the high-order part of the ACBXISQG field to binary zeros.

As a result of an initial S*x* call, Adabas returns the number of records that satisfy the search criterion in this field. If security-by-value is being used, response code 1 (ADARSP001) is returned in this field along with the value 0 (one record found) or 1 (more than one record found). For more information, see the *Adabas Security Manual*.

As a result of a subsequent S*x* call used to retrieve ISNs from Adabas Work, Adabas provides the number of returned ISNs in this field. The ISNs themselves are returned in the ISN buffer.

Command Option 1 (ACBXCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S*x* command on Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
R (return)	For an S4 command, returns response code 145 (ADARSP145) if a record to be read and held is not available.

Command Option 2 (ACBXCOP2)

Option	Description
D (descending sequence)	For an S2 command, sorts descriptor values in descending sequence.

If no Command Option 2 is specified for an S2 command, the descriptor values are sorted in ascending sequence.

Command Option 1/2: Release CID Option (ACBXCOP1 and ACBXCOP2)

The "I" option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field as the first action taken during command execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Command Option 3: Shared Hold Status (ACBXCOP3)

The following command options are available for the S4 command only:

Option	Description
C	Puts the record in shared hold status for the duration of the read operation.
Q	Puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first. A command ID must be given and the ISN buffer length must be set to 4. The record returned by the S4 remains in shared hold until the next record is retrieved by a subsequent S4 or L4/N command with the same command ID.

Option	Description
S	Puts the record in shared hold status until the end of the transaction.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For complete information about shared hold updating, read ***Shared Hold Status***, elsewhere in this guide.

Additions 1: S2 Command, Descriptors Used for Sort Control (ACBXADD1)

If the S2 command is being used, this field must specify the descriptor (or descriptors) to be used to control the sort sequence; if no sort argument is specified, the S2 command returns response code 28 (ADARSP028).

One to three descriptors, including subdescriptors and superdescriptors, can be specified. Phonetic descriptors or descriptors contained within a periodic group cannot be specified. A multiple-value field can be specified, in which case the ISNs will be sorted according to the lowest value present within a given record.

Any unused positions of this field must be set to blanks. For example:

```
XXYYbbbb
```

where:

XX is the major sort descriptor; and

YY is the minor sort descriptor

bbbb blanks

The number of ISNs that can be sorted depends on the size of the sort work area (ADARUN LS parameter) defined by the DBA. If the sort area is too small, no sort will be performed; response code 1 (ADARSP001) will be returned, and the ISNs will be returned in ascending sequence.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Cipher Code and Version/Nucleus ID (ACBXADD4)

This field is used to provide a cipher code. If the file is ciphered, the user must provide a valid cipher code. If the file is not ciphered, this field should be set to blanks.

Adabas sets any cipher code to blanks during command processing, and returns a version code and the database ID in the rightmost (low-order) three bytes of this field. For more information, see the section ***Control Block Fields***.

Additions 5: Format ID, Global Format ID (ACBXADD5)

This field may be used to provide a separate format ID to identify the internal format buffer to be used for this command, or to provide a global format ID.

If the high-order bit of the Additions 5 field is zero (0), the value provided in the command ID field is also used as the format ID.

If, however, this bit is set to 1, the fifth through eighth bytes of the Additions 5 field are used as the format ID.

If the two high-order (leftmost) bits of the first byte of Additions 5 field are set to one (B'11'), all eight bytes of the Additions 5 field are used as a global format ID (that is, the format ID can be used by several users at the same time).

For more information, refer to the section ***Command, Format, and Global Format IDs***.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Compressed Record Length (ACBXLCMP)

This field returns the compressed record length when a record was read or written. This is the length of the compressed data processed by the successful Adabas call. If the logical data storage record spans multiple physical data records, the combined length of all associated physical records may not be known. In this case, Adabas returns high values in the low-order word of this field.

Decompressed Record Length (ACBXLDEC)

This field returns the decompressed record length. This is the length of the decompressed data processed by the successful call. If multiple record buffer segments are specified, this reflects the total length across all buffer segments.

Buffers

The following buffers should be specified with the S1, S2, and S4 commands:

- Format Buffer
- Record Buffer
- Search and Value Buffers

- ISN Buffer

Format Buffer

If the record identified by the first ISN in the resulting ISN list is to be read from Data Storage, the fields within the record for which values are to be returned must be specified in this buffer. Descriptions of the syntax and examples of format buffer construction are provided in ***Defining Buffers***, elsewhere in this guide. User-specified fields for controlling the S2 command's ISN sort sequence must be specified in the Additions 1 field.

If no read is to be performed, the first non-blank character of this buffer must be a period (.).

If a valid command ID is specified, Adabas retains this decoded format buffer for use by later commands specifying the same command ID.

Record Buffer

If the format buffer contains field definitions to enable the read option, Adabas returns the requested field values in this buffer.

The values are returned according to the standard length and format of the field, unless the user specifies a different length and/or format in the format buffer.

Search and Value Buffers

Search and value buffers are used to define the search criteria. The search expression (or expressions) is provided in the search buffer, and the values which correspond to the search expressions are provided in the value buffer.

Descriptions of the syntax and examples of format buffer construction are provided in ***Defining Buffers***, elsewhere in this guide.

ISN Buffer

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. The ISNs are returned in ascending ISN sequence unless the S2 command is being used, in which case they are returned in the user-specified sort sequence.

If the query contains one or more file-coupling criteria, the resulting ISN list contains only those ISNs in the primary file (the file specified in the control block's file number field).

If the ISN buffer length field is set to less than 4, no ISNs are returned in the ISN buffer. If a valid command ID is specified, the ISN buffer length is not zero, but the ISN buffer is still too small to contain all the resulting ISNs, Adabas will store the overflow ISNs on Adabas Work. These ISNs may then be retrieved using further S1, S2, or S4 calls in which the same command ID is used.

See the section ***ISN List Processing*** for additional information.

42 S5 Command: Find Coupled ISNs

The S5 command returns or saves a list of coupled ISNs for the specified file.

Function and Use

The S5 command is used to determine the records in one file that are coupled to a given record in another file.

The user specifies the file number and a given ISN within the file, plus the file number from which the coupled ISNs are to be returned. An optional ISN value above which ISNs are to be returned can also be specified.

Adabas determines which records are coupled to the specified record by using the Associator coupling lists. No access to Data Storage is required.

Adabas returns the resulting ISNs in the **ISN buffer**.

ACB Interface Direct Call: S5 Command

This section describes ACB interface direct calls for the S5 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Overview

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	F	A
ISN Lower Limit	17-20	binary	F	U
ISN Quantity	21-24	binary	--	A
	25-32	--	--	--

Field	Position	Format	Before Adabas Call	After Adabas Call
ISN Buffer Length	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
	45-48	--	--	--
Additions 3	49-56	alphanumeric	F	A
	57-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	--
Record	*	--
Search	*	--
Value	*	--
ISN	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

S5

Command ID (ACBCID)

A non-blank, non-zero value can be specified in this field if the Save ISN List option is to be used, or if overflow ISNs are to be stored on and then later read from the Adabas Work.

The first byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

The number of the file from which the coupled ISNs are to be selected. This file, called the *primary file*, must be coupled to the file specified in the Additions 1 field, and cannot be an Adabas expanded file. The file number of physically coupled files must be no greater than 255.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Nonzero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

The ISN of the record for which the coupled ISNs are to be returned. The ISN must be present in the file specified in the Additions 1 field. Adabas will return the first ISN of the resulting coupled ISN list in this field.

ISN Lower Limit (ACBISL)

This field may be used in an initial S*x* call to limit the resulting ISN list to those ISNs which are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work.

ISN Quantity (ACBISQ)

An initial S5 call returns the number of records in the specified file that satisfy the search criteria.

In subsequent S*x* calls used to retrieve ISNs from the Adabas Work, this field contains the number of ISNs placed in the ISN buffer.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs will be inserted in the ISN buffer.

To save the ISN list on Work for later processing, specify "H" in the Command Option 1 field and a valid command ID. The ISN buffer length field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer which is a multiple of 4.

Command Option 1: Save ISN List Option (ACBCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S5 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

Command Option 1 or 2: Release Command ID Option (ACBCOP1 or ACBCOP2)

The "I" option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field. This is the first action taken during S5 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Additions 1: File Number (ACBADD1)

The number of the file which contains the ISN specified in the ISN field. The file number must be entered in the first two bytes of this field. The number must be provided in binary format. The remaining positions of this field must be set to blanks. This field must not be changed between successive S5 calls.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

ACB Example

Select the records in file 2 that are coupled to the record in file 1 identified with ISN 5. Use the save-ISN-list option.

Control Block

Command Code	S5	
Command ID	S501	a non-blank command ID is required if the save-ISN-list option is to be used
File Number	2	records to be selected from file 2
ISN	5	ISN 5 identifies the record for which the coupled records are to be selected
ISN Lower Limit	0	all qualifying ISNs are to be selected
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	save-ISN-list option to be used

Additions 1	X'0001404040404040'	the record for which the coupled records are to be selected is contained in file 1
Additions 3	password	file is security-protected

ACBX Interface Direct Call: S5 Command

This section describes ACBX interface direct calls for the S5 command. It covers the following topics:

- Control Block and Buffer Overview
- Control Block Field Descriptions

Control Block and Buffer Overview

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	F	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
	65-68	---	---	---

Field	Position	Format	Before Adabas Call	After Adabas Call
Additions 3	69-76	alphanumeric/ binary	F	A
	77-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
ISN	---	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
S5

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)
A non-blank, non-zero value can be specified in this field if the Save ISN List option is to be used, or if overflow ISNs are to be stored on and then later read from the Adabas Work.

The first byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file from which the coupled ISNs are to be selected. This file, called the *primary file*, must be coupled to the file specified in the Additions 1 field, and cannot be an Adabas expanded file. The file number of physically coupled files must be no greater than 255.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

The ISN of the record for which the coupled ISNs are to be returned. The ISN must be present in the file specified in the Additions 1 field. Adabas will return the first ISN of the resulting coupled ISN list in this field.

The ACBXISN field is a four-byte binary field embedded in the eight-byte ACBXISNG field, which is not yet used. Set the high-order part of the ACBXISNG field to binary zeros.

ISN Lower Limit (ACBXISL)

This field may be used in an initial S*x* call to limit the resulting ISN list to those ISNs which are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work.

ISN Quantity (ACBXISQ)

An initial S5 call returns the number of records in the specified file that satisfy the search criteria.

In subsequent S*x* calls used to retrieve ISNs from the Adabas Work, this field contains the number of ISNs placed in the ISN buffer.

Command Option 1: Save ISN List Option (ACBXCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S5 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

Command Option 1 or 2: Release Command ID Option (ACBXCOP1 or ACBXCOP2)

The "I" option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field. This is the first action taken during S5 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Additions 1: File Number (ACBXADD1)

The number of the file which contains the ISN specified in the ISN field. The file number must be entered in the first two bytes of this field. The number must be provided in binary format. The remaining positions of this field must be set to blanks. This field must not be changed between successive S5 calls.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to the S5 command:

- Format Buffer
- Record Buffer
- Search Buffer
- Value Buffer

- ISN Buffer

Format Buffer

If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a format buffer is not needed.

Record Buffer

If this is an **ACB interface direct call** and a record buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a record buffer is not needed.

Search Buffer

If this is an **ACB interface direct call** and a search buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a search buffer is not needed.

Value Buffer

If this is an **ACB interface direct call** and a value buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a value buffer is not needed.

ISN Buffer

Adabas places the list of resulting ISNs in the ISN buffer. Each ISN is returned as a four-byte binary number. The first ISN in the list is also returned in the control block's ISN field.

The ISNs are returned in ascending ISN sequence.

If the ISN buffer length is neither zero nor large enough to contain all the resulting ISNs, and a valid command ID was used, Adabas will store the overflow ISNs on the Adabas Work. These ISNs may then be retrieved using additional S5 calls that specify the same command ID. See the section ***ISN List Processing*** for additional information.

43 S8 Command: Process ISN Lists

The S8 command combines two ISN lists from the same file with an AND, OR, or NOT operation. For more information refer to the section *ISN List Processing*.

Function and Use

The S8 command performs logical processing on two ISN lists that were previously created with S*x* commands. Both ISN lists *must* be

- derived from the same file;
- in ISN sequence;
- stored on the Work data set; and
- identified by command IDs assigned to the lists when they were created.

ISN lists resulting from an S2 or S9 command that are not in ascending ISN sequence cannot be used.

No activity (access or update) may be performed on the ISN lists to be processed between the time they are created and the time the S8 command is executed.

The S8 command may be used to perform the following logical operations:

Operator	The resulting ISN list contains those ISNS that are present in . . .
AND	both ISN lists
OR	either of the ISN lists
NOT	the first ISN list but not the second ISN list

The resulting ISNs are returned in the ISN buffer and/or stored on the Work data set in ascending ISN sequence, depending on the specified Command Option and the command ID field setting:

- The resulting ISN list is saved in both the ISN buffer and on the Work data set when a non-blank, non-zero command ID is specified and the save-ISN-list option is also specified.
- The resulting ISN list is saved in the ISN buffer *but not* on the Work data set when no (or an invalid) command ID is specified with the save-ISN-list option.

ACB Interface Direct Call: S8 Command

This section describes ACB interface direct calls for the S8 command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions
- ACB Example

Control Block and Buffer Information

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	F	U
ISN Quantity	21-24	binary	--	A
	25-32	--	--	--
ISN Buffer Length	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
	45-48	--	--	--
Additions 3	49-56	alphanumeric	F	A
	57-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	--
Record	*	--
Search	*	--
Value	*	--
ISN	--	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement

Control Block Field Descriptions

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)

S8

Command ID (ACBCID)

A nonblank, nonzero command ID must be specified in this field if a command option is specified in Command Option 1 field:

- The i (release command ID) option releases the specified command ID and any related ISN list as the first action taken during the S8 execution.
- With the H (save-ISN-list) option, the ISN list resulting from the S8 execution is stored under the specified command ID. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

For more information refer to the section *ISN List Processing*.

The first byte of this field may not be set to hexadecimal 'FF'.

File Number (ACBFNR)

The number of the file from which both ISN lists to be processed were obtained.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work data set.

ISN Lower Limit (ACBISL)

This field may be used in an initial S*x* call to limit the resulting ISN list to those ISNs that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work data set.

ISN Quantity (ACBISQ)

As a result of an initial S8 call, this field returns the number of ISNs in the resulting ISN list.

As a result of a subsequent S8 call to retrieve ISNs from the Adabas Work data set, this field contains the number of ISNs returned in the ISN buffer.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes). This length is used to determine the number of ISNs placed in the ISN buffer.

If this field is set to zeros, no ISNs are inserted in the ISN buffer. This field should be set to zeros if the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command, or if the command is being issued only to determine the number of qualifying records.

If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer which is a multiple of 4.

Command Option 1: Save ISN List Option, Release Command ID Option (ACBCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S8 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
I	Releases the command ID (CID) value specified in the command ID field and any related ISN list as the first action taken during the S8 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Command Option 2: Logical Operator (ACBCOP2)

The value entered in this field indicates the logical operation to be performed on the ISN lists:

Option	Operation	The resulting ISN list contains those ISNs that are present in ...
D	AND	both ISN lists.
O	OR	either ISN list.
N	NOT	the first ISN list but not the second ISN list.

Additions 1: Command IDs (ACBADD1)

The command IDs that identify the ISN lists to be processed must be specified in this field (four bytes per command ID). Each ISN list must be currently stored on the Adabas Work data set and should contain ISNs from the same file.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

ACB Example

Perform a logical OR operation between two ISN lists to produce a third ISN list that contains ISNs present in either list. The ISN lists to be processed were stored on the Adabas Work data set under the command IDs "U020" and "U021". Store the resulting ISN list on the Adabas Work under the command ID "U999". Use the save-ISN-list option.

Control Block

Command Code	S8	
Command ID	U999	store the resulting ISN list under the command ID U999
ISN Lower Limit	0	select all of the resulting ISNs
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	use the save-ISN-list option
Command Option 2	O	perform an OR operation
Additions 1	U020U021	process the ISN lists identified by the command IDs U020 and U021
Additions 3	*bbbbbbbb* (blanks)	file not security-protected

ACBX Interface Direct Call: S8 Command

This section describes ACBX interface direct calls for the S8 command. It covers the following topics:

- Control Block and Buffer Information

- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-28	---	---	---
ISN	29-32	binary	---	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-44	---	---	---
ISN Quantity	45-48	binary	---	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
	65-68	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
	77-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
ISN	---	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

S8

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

A nonblank, nonzero command ID must be specified in this field if a command option is specified in Command Option 1 field:

- The "I" (release command ID) option releases the specified command ID and any related ISN list as the first action taken during the S8 execution.
- With the "H" (save-ISN-list) option, the ISN list resulting from the S8 execution is stored under the specified command ID. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.

For more information refer to the section *ISN List Processing*.

The first byte of this field may not be set to hexadecimal 'FF'.

Database ID (ACBXDBID)

Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

Use this field to specify the number of the file from which both ISN lists to be processed were obtained.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

ISN (ACBXISN)

Adabas returns the first ISN of the resulting ISN list in this field. If there were no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work data set.

ISN Lower Limit (ACBXISL)

This field may be used in an initial S*x* call to limit the resulting ISN list to those ISNs that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

This field is also used when a group of ISNs from a saved ISN list is being retrieved from the Adabas Work data set.

ISN Quantity (ACBXISQ)

As a result of an initial S8 call, this field returns the number of ISNs in the resulting ISN list.

As a result of a subsequent S8 call to retrieve ISNs from the Adabas Work data set, this field contains the number of ISNs returned in the ISN buffer.

Command Option 1: Save ISN List Option, Release Command ID Option (ACBXCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S8 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost.
I	Releases the command ID (CID) value specified in the command ID field and any related ISN list as the first action taken during the S8 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs.

Command Option 2: Logical Operator (ACBXCOP2)

The value entered in this field indicates the logical operation to be performed on the ISN lists:

Option	Operation	The resulting ISN list contains those ISNs that are present in ...
D	AND	both ISN lists.
O	OR	either ISN list.
N	NOT	the first ISN list but not the second ISN list.

Additions 1: Command IDs (ACBXADD1)

The command IDs that identify the ISN lists to be processed must be specified in this field (four bytes per command ID). Each ISN list must be currently stored on the Adabas Work data set and should contain ISNs from the same file.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to the S8 command:

- Format Buffer
- Record Buffer
- Search Buffer
- Value Buffer
- ISN Buffer

Format Buffer

If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a format buffer is not needed.

Record Buffer

If this is an **ACB interface direct call** and a record buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a record buffer is not needed.

Search Buffer

If this is an **ACB interface direct call** and a search buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a search buffer is not needed.

Value Buffer

If this is an **ACB interface direct call** and a value buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a value buffer is not needed.

ISN Buffer

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. The ISNs are returned in ISN sequence.

If the ISN buffer is too small to contain all the resulting ISNs and a non-blank, non-zero command ID was used, Adabas stores the overflow ISNs on the Work data set. These ISNs can be retrieved with further S*x* calls using the same command ID. For more information refer to the section ***ISN List Processing***.

44 S9 Command: Sort ISN Lists

The S9 command sorts an ISN list in ascending ISN or descriptor-specified sequence.

Function and Use

The S9 command sorts an ISN list provided by the user or created by a previous S*x* command. The ISN list to be sorted may either be in the **ISN buffer** or on the Adabas Work data set (if the command ID assigned to the list when it was created is specified in the Additions 4 field).

The ISN list may be sorted in order of:

- ISN value (ascending ISN sequence);
- one to three user-specified descriptors.

  You can specify from one to three descriptors which are to be used to control the sort sequence. Either ascending or descending sequence may be specified.

The ISN list to be sorted must contain ISNs in ascending sequence, which implies that the list was not created by an S2 or an S9 command that specified the descriptor sequence option.

The resulting ISN list is either returned in the **ISN buffer** or, optionally, stored on the Adabas Work data set.

The S9 command can also be performed on Adabas expanded files.

ACB Interface Direct Call: S9 Command

This section describes ACB interface direct calls for the S9 command. It covers the following topics:

Control Block and Buffer Information

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	--	--	--
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	binary	F	U
Response Code	11-12	binary	--	A
ISN	13-16	binary	--	A
ISN Lower Limit	17-20	binary	F	U
ISN Quantity	21-24	binary	F	A
	25-32	--	--	--
ISN Buffer Length	33-34	binary	F	U
Command Option 1	35	alphanumeric	F	U
Command Option 2	36	alphanumeric	F	U
Additions 1	37-44	alphanumeric	F	U
	45-48	--	--	--
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	F	A
	65-72	--	--	--
Command Time	73-76	binary	--	A
User Area	77-80	--	--	U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	*	--
Record	*	--
Search	*	--
Value	*	--
ISN	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
*	Not used but must be included in parameter list of call statement
--	Not used

Control Block

We recommend that you set unused ACB fields to binary zeros before the direct call is initiated.

Command Code (ACBCMD)
S9

Command ID (ACBCID)
A non-blank, non-zero command ID may be specified in this field. This command ID applies only to the "I" (release command ID) or "H" (save-ISN-list) options; the command ID for obtaining an ISN list stored on Work must be specified in the Additions 4 field.

The first byte of this field may not be set to a hexadecimal 'FF'.

File Number (ACBFNR)
The number of the file from which the ISN list to be sorted was obtained.

The S9 command can also be performed on Adabas expanded files.

Note: When using two-byte file numbers and database IDs, a X'30' must be coded in the first byte of the control block.

Response Code (ACBRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes Manual* documentation.

ISN (ACBISN)
Adabas returns the first ISN of the resulting ISN list in this field. If there are no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work data set.

ISN Lower Limit (ACBISL)
This field may be used in an initial S*x* call to limit the resulting ISN list to those that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

This field is also used when a group of overflow ISNs from a saved ISN list is being retrieved from the Adabas Work data set.

ISN Quantity (ACBISQ)
If the ISN list to be sorted is being provided in the ISN buffer, this field must contain the number of ISNs that are to be sorted. If security-by-value is being used, a response code 1 (ADARSP001) is returned combined with the value 0 (one record found) or 1 (more than one record found) in this field. For more information, see the *Adabas Security Manual*.

For an initial S9 call, Adabas returns the number of records contained in the resulting ISN list. For any subsequent S9 call that retrieves ISNs from the Work data set, Adabas returns the number of ISNs placed in the ISN buffer.

ISN Buffer Length (ACBIBL)

The ISN buffer length (in bytes) is used to determine the number of ISNs placed in the ISN buffer.

- If this field is set to zeros, no ISNs are inserted in the ISN buffer. Set this field to zeros and specify "H" in the Command Option 1 field if the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command. If the S9 command is being issued only to determine the number of qualifying records, specify zero in this field and no command ID to prevent a sorted ISN list from being returned or stored.
- If a non-zero value is specified, it should be a multiple of 4. If it is not, Adabas reduces the length to the next lower integer that is a multiple of 4.

If the ISNs to be sorted are contained in the ISN buffer, this field must contain a value equal to or larger than the number of ISNs to be sorted, multiplied by 4. ISN overflow is stored on Work, and can be retrieved by a later S*x* command with a command ID matching that specified in the command ID field.

Command Option 1: Save ISN List Option (ACBCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S9 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost. If the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command, use this option with the ISN buffer length field set to zeros. If this option is specified, Command Option 2 *cannot* specify the "I" (release command ID) option.

Command Option 2: Descending Option (ACBCOP2)

Option	Description
D (descending sequence)	Sorts the ISN list in descending sequence. This option may *not* be specified when sorting by ISN values.

If no Command Option 2 is specified for an S9 command, the ISN list is sorted in ascending sequence.

Command Option 1 or 2: Release Command ID Option (ACBCOP1 or ACBCOP2)

The "I" option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field as the first action taken during S9 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs. If the "H" (save-ISN-list) option is specified as Command Option 1, this option *cannot* be specified as Command Option 2.

Additions 1: Sort Sequence (ACBADD1)

The sort sequence to be used must be specified in this field.

The value "ISN*bbbbb*" indicates that the ISN values are to be used as the sorting sequence (where *bbbbb* represents blanks).

If the sort sequence is to be based on the values of one or more descriptors, the descriptors to be used must be specified in this field. One to three descriptors, subdescriptors, or superdescriptors may be specified. Phonetic descriptors and descriptors contained within a periodic group may not be specified. A multiple-value field may be specified, in which case the ISNs will be sorted according to the lowest value present within a given record. The descriptors are specified beginning with byte 1 (left justified). Any remaining positions must be set to blanks.

The number of ISNs that can be sorted depends on the size of the ADARUN parameters defined by the DBA. If this limit is exceeded, no sort is performed and response code 1 (ADARSP001) is returned.

Additions 3: Password (ACBADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Command ID (ACBADD4)

If the ISN list to be sorted is contained on the Adabas Work data set, the command ID under which the list is stored must be specified in the first 4 bytes of this field.

If the ISN list to be sorted is in the ISN buffer, this field must be set to blanks.

Adabas sets the Additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

ACB Example

- Example 1
- Example 2

Example 1

Sort an ISN list contained in the ISN buffer in ISN sequence. Sort 622 ISNs.

Control Block

Command Code	S9	
Command ID	S901	a non-blank, non-zero command ID is required
File Number	1	derive the ISN list to be sorted from file 1
ISN Quantity	622	sort 622 ISNs
ISN Lower Limit	0	select all ISNs
ISN Buffer Length	2488	or larger; each ISN to be sorted requires 4 bytes
Command Option 1	H	use the save-ISN-list option
Command Option 2	*b* (blank)	use ascending sequence
Additions 1	ISN*bbbbb*	use the ISN values as the sorting sequence, where *bbbbb* represents blanks.
Additions 3	*bbbbbbbb* (blanks)	file not security-protected
Additions 4	*bbbbbbbb* (blanks)	the ISN list to be sorted is contained in the ISN buffer

Buffer Areas

ISN Buffer	The ISNs to be sorted are provided in this buffer. Each ISN must be provided as a 4-byte binary number.

Example 2

Sort an ISN list stored on the Adabas Work. The command ID under which the ISN list is stored is "U066". Sort the list using the descriptors AA and AB as the major and minor sequence fields. Use the descending option.

Control Block

Command Code	S9	
Command ID	S902	a non-blank, non-zero command ID is required
File Number	1	derive the ISN list to be sorted from file 1
ISN Lower Limit	0	select all ISNs
ISN Buffer Length	0	no ISNs are to be returned in the ISN buffer
Command Option 1	H	use the save-ISN-list option
Command Option 2	D	use the descending sequence
Additions 1	AAAB*bbbb*	use AA as the major sequence field and AB as the minor sequence field, where *bbbb* represents blanks.
Additions 3	*bbbbbbbb* (blanks)	file not security-protected
Additions 4	U066*bbbb*	the ISN list to be sorted is stored on the Adabas Work under the command ID "U066", where *bbbb* represents blanks.

ACBX Interface Direct Call: S9 Command

This section describes ACBX interface direct calls for the S9 command. It covers the following topics:

- Control Block and Buffer Information
- Control Block Field Descriptions

Control Block and Buffer Information

Control Block

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	---	---	---
Version Indicator	3-4	binary	F	U
	5-6	---	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	---	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U

Field	Position	Format	Before Adabas Call	After Adabas Call
	25-28	---	---	---
ISN	29-32	binary	---	A
	33-36	---	---	---
ISN Lower Limit	37-40	binary	F	U
	41-44	---	---	---
ISN Quantity	45-48	binary	F	A
Command Option 1	49	alphanumeric	F	U
Command Option 2	50	alphanumeric	F	U
	51-56	---	---	---
Additions 1	57-64	alphanumeric/ binary	F	U
	65-68	---	---	---
Additions 3	69-76	alphanumeric/ binary	F	A
Additions 4	77-84	alphanumeric	F	A
	85-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
---	169-193	do not touch	---	---

ABDs and Buffers

ABD and Buffer	Before Adabas Call	After Adabas Call
ISN	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call
---	Not used

Control Block Field Descriptions

We recommend that you set unused ACBX fields to binary zeros before the direct call is initiated.

Version Indicator (ACBXVER)
F2

Command Code (ACBXCMD)
S9

Response Code (ACBXRSP)
Adabas returns the response code for the command in this field. Response code 0 (ADARSP000) indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)
A non-blank, non-zero command ID may be specified in this field. This command ID applies only to the "I" (release command ID) or "H" (save-ISN-list) options; the command ID for obtaining an ISN list stored on Work must be specified in the Additions 4 field.

The first byte of this field may not be set to a hexadecimal 'FF'.

Database ID (ACBXDBID)
Use this field to specify the database ID. The Adabas call will be directed to this database.

This field is a four-byte binary field, but at this time only two-byte database IDs are supported. Therefore, the database ID should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)
Use this field to specify the number of the file from which the ISN list to be sorted is obtained.

This field is a four-byte binary field, but the file number should be specified in the low-order part (rightmost bytes) of the field, with leading binary zeros.

The S9 command can also be performed on Adabas expanded files.

ISN (ACBXISN)
Adabas returns the first ISN of the resulting ISN list in this field. If there are no resulting ISNs, this field is not modified. This applies to both the initial call and any subsequent calls that are used to retrieve ISNs from the Adabas Work data set.

ISN Lower Limit (ACBXISL)
This field may be used in an initial S*x* call to limit the resulting ISN list to those that are greater than the ISN specified in this field. If this field is set to zeros, Adabas returns all qualifying ISNs.

The ACBXISL field is a four-byte binary field embedded in the eight-byte ACBXISLG field, which is not yet used. Set the high-order part of the ACBXISLG field to binary zeros.

This field is also used when a group of overflow ISNs from a saved ISN list is being retrieved from the Adabas Work data set.

ISN Quantity (ACBXISQ)

If the ISN list to be sorted is being provided in the ISN buffer, this field must contain the number of ISNs that are to be sorted. If security-by-value is being used, a response code 1 (ADARSP001) is returned combined with the value 0 (one record found) or 1 (more than one record found) in this field. For more information, see the *Adabas Security Manual*.

The ACBXISQ field is a four-byte binary field embedded in the eight-byte ACBXISQG field, which is not yet used. Set the high-order part of the ACBXISQG field to binary zeros.

For an initial S9 call, Adabas returns the number of records contained in the resulting ISN list. For any subsequent S9 call that retrieves ISNs from the Work data set, Adabas returns the number of ISNs placed in the ISN buffer.

Command Option 1: Save ISN List Option (ACBXCOP1)

Option	Description
H (save ISN list)	Stores the entire ISN list resulting from an S9 command on the Adabas Work under the specified command ID. A valid command ID must be specified. If no command ID is specified, the ISN list is not stored on Work and any ISNs not saved in the ISN buffer are lost. If the resulting ISN list is to be read with the GET NEXT option of the L1 or L4 command, use this option with the ISN buffer length field set to zeros. If this option is specified, Command Option 2 *cannot* specify the "I" (release command ID) option.

Command Option 2: Descending Option (ACBXCOP2)

Option	Description
D (descending sequence	Sorts the ISN list in descending sequence. This option may *not* be specified when sorting by ISN values.

If no Command Option 2 is specified for an S9 command, the ISN list is sorted in ascending sequence.

Command Option 1 or 2: Release Command ID Option (ACBXCOP1 and ACBXCOP2)

The "I" option may be specified in either the Command Option 1 or Command Option 2 field:

Option	Description
I	Releases the command ID (CID) value specified in the command ID field as the first action taken during S9 execution. The specified command ID is released *only* from the table of ISN lists. The same command ID is then reused to identify the resulting list of ISNs. If the "H" (save-ISN-list) option is specified as Command Option 1, this option *cannot* be specified as Command Option 2.

Additions 1: Sort Sequence (ACBXADD1)

The sort sequence to be used must be specified in this field.

The value "ISN*bbbbb*" indicates that the ISN values are to be used as the sorting sequence, where *bbbbb* represents blanks.

If the sort sequence is to be based on the values of one or more descriptors, the descriptors to be used must be specified in this field. One to three descriptors, subdescriptors, or superdescriptors may be specified. Phonetic descriptors and descriptors contained within a periodic group may not be specified. A multiple-value field may be specified, in which case the ISNs will be sorted according to the lowest value present within a given record. The descriptors are specified beginning with byte 1 (left justified). Any remaining positions must be set to blanks.

The number of ISNs that can be sorted depends on the size of the ADARUN parameters defined by the DBA. If this limit is exceeded, no sort is performed and response code 1 (ADARSP001) is returned.

Additions 3: Password (ACBXADD3)

This field is used to provide an Adabas security password. If the database, file, or fields are security-protected, the user must provide a valid security password. Adabas sets the Additions 3 field to blanks during command processing to enhance password integrity.

Additions 4: Command ID (ACBXADD4)

If the ISN list to be sorted is contained on the Adabas Work data set, the command ID under which the list is stored must be specified in the first 4 bytes of this field.

If the ISN list to be sorted is in the ISN buffer, this field must be set to blanks.

Adabas sets the Additions 4 field to blanks during command processing, and returns a version code and database ID in the rightmost (low-order) three bytes of this field. For more information, see the section *Control Block Fields*.

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to the S9 command:

- Format Buffer
- Record Buffer
- Search Buffer
- Value Buffer
- ISN Buffer

Format Buffer

If this is an **ACB interface direct call** and a format buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a format buffer is not needed.

Record Buffer

If this is an **ACB interface direct call** and a record buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a record buffer is not needed.

Search Buffer

If this is an **ACB interface direct call** and a search buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a search buffer is not needed.

Value Buffer

If this is an **ACB interface direct call** and a value buffer is not specified, a processing error will occur; ACB interface direct calls expect buffers to be specified in a set sequence. If this is an **ACBX interface direct call**, a value buffer is not needed.

ISN Buffer

The ISN list to be sorted may be provided by the user in this buffer.

Adabas places the list of resulting ISNs in this buffer. Each ISN is returned as a four-byte binary number. If the ISN sort option is in effect ("ISN*bbbbb*" is entered in the Additions 1 field), the ISNs are returned in ascending ISN sequence. Otherwise, the ISNs are returned in the order of the values of the user-specified descriptors.

If the ISN buffer length is neither zero nor large enough to contain all the resulting ISNs and a valid command ID was specified, Adabas stores the overflow ISNs on the Adabas Work data set. These ISNs may then be retrieved using further S9 calls in which the same command ID is specified in the Additions 4 field. For more information, refer to the section ***ISN List Processing***, elsewhere in this guide.

VII Programming Examples

This chapter provides examples of direct Adabas calls in a variety of languages.

The information is organized under the following headings:

45 ACB Examples

This chapter describes examples of direct calls using the ACB interface.

File Definitions Used in ACB Examples

The following file definitions are used in the examples that follow in this documentation. Two file structures (files 1 and 2) are used.

The following tables show the structures of files 1 and 2 where

SF	is standard format
SL	is standard length

File 1

File 1 is neither security-protected nor ciphered. Certain examples in the documentation assume that file 1 has been coupled to file 2 using the descriptor AA as the basis for coupling.

Adabas Definition	Explanation
01,GA	Group GA, consisting of fields AA and AB.
02,AA,8,A,DE,NU	Elementary field AA; SL is 8 bytes, SF is alphanumeric, descriptor, null value suppression.
02,AB,2,P,DE,NU	Elementary field AB; SL is 2, SF is packed, descriptor, null value suppression.
01,AC,20,A,NU	Elementary field AC; SL is 20, SF is alphanumeric, null value suppression.
01,MF,3,A,MU,DE,NU	Multiple value field MF; SL is 3, SF is alphanumeric, descriptor, null value suppression.
01,GB,PE	Periodic group GB.
02,BA,1,B,DE,NU	Elementary field BA (within periodic group GB); SL is 1, SF is binary, descriptor, null value suppression.
02,BB,5,P,NU	Elementary field BB (within periodic group GB); SL is 5, SF is packed, null value suppression.
02,BC,10,A,NU	Elementary field BC (within periodic group GB); SL is 10, SF is alphanumeric, null value suppression.
01,GC,PE	Periodic group GC.
02,CA,7,A,DE,NU	Elementary field CA (within periodic group GC); SL is 7, SF is alphanumeric, descriptor, null value suppression.
02,CB,10,A,MU,NU SF	Multiple value field CB (within periodic group GC); SL is 10, is alphanumeric, null value suppression.

File 2

File 2 is security-protected. It is not ciphered. Certain examples in this documentation assume that file 2 is coupled to file 1 using field RA as the basis for coupling.

Adabas Definition	Explanation
01,RG	Group RG, consisting of all the fields in the record.
02,RA,8,A,DE,NU	Elementary field RA; SL is 8, SF is alphanumeric, descriptor, null value suppression.
02,RB,10,A,DE	Elementary field RB; SL is 10, SF is alphanumeric, descriptor.
02,GX	Group GX, consisting of the fields XA, XB, XC, XD, and XE.
03,XA,10,A	Elementary field XA; SL is 10, SF is alphanumeric.
03,XB,2,P,DE	Elementary field XB; SL is 2, SF is packed, descriptor.
03,XC,6,U	Elementary field XC; SL is 6, SF is unpacked.
03,XD,8,A,DE,NU	Elementary field XD; SL is 8, SF is alphanumeric, descriptor, null value suppression.
03,XE,5,A,DE,NU	Elementary field XE; SL is 5, SF is alphanumeric, descriptor, null value suppression.
SA=RA(1,4)	Subdescriptor SA; derived from bytes 1 through 4 of field RA, format is alphanumeric.
SB=RA(1,8),RB(1,4)	Superdescriptor SB; derived from bytes 1 through 8 of field RA and bytes 1 through 4 of field RB, format is alphanumeric.
SC=XB(1,2),XC(1,6)	Superdescriptor SC; derived from bytes 1 through 2 of field XB and bytes 1 through 6 of field XC, format is binary.

ACB Assembler Examples

This section contains examples of using direct Adabas calls in Assembler. The **previously defined Adabas files** defined are used in each example.

```
*** CONTROL BLOCK
      DS    0F
CB    DS    0CL80         USER CONTROL BLOCK
      DC    CL2' '        RESERVED FOR ADABAS USE
CCODE DC    CL2' '        COMMAND CODE
CID   DC    CL4' '        COMMAND ID
FNR   DC    H'0'          FILE NUMBER
RC    DC    H'0'          RESPONSE CODE
ISN   DC    F'0'          ISN
ISNLL DC    F'0'          ISN LOWER LIMIT
ISNQ  DC    F'0'          ISN QUANTITY
FBL   DC    H'100'        FORMAT BUFFER LENGTH
RBL   DC    H'250'        RECORD BUFFER LENGTH
SBL   DC    H'50'         SEARCH BUFFER LENGTH
VBL   DC    H'100'        VALUE BUFFER LENGTH
IBL   DC    H'20'         ISN BUFFER LENGTH
COPT1 DC    CL1' '        COMMAND OPTION 1
```

```
COPT2 DC   CL1' '       COMMAND OPTION 2
ADD1  DC   CL8' '       ADDITIONS 1
ADD2  DC   CL4' '       ADDITIONS 2
ADD3  DC   CL8' '       ADDITIONS 3
ADD4  DC   CL8' '       ADDITIONS 4
ADD5  DC   CL8' '       ADDITIONS 5
CTIME DC   F'0'         COMMAND TIME
UAREA DC   CL4' '       USER AREA
*
*
*** USER BUFFER AREAS
FB    DC   CL100' '     FORMAT BUFFER
RB    DC   CL250' '     RECORD BUFFER
SB    DC   CL50' '      SEARCH BUFFER
VB    DC   CL100' '     VALUE BUFFER
IB    DC   CL20' '      ISN BUFFER
* * * ↩
```

This section provides the following examples:

- Example 1
- Example 2
- Example 3 : User Session with ET Logic

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
EXMP1 MVC  CCODE,=C'OP'             OP COMMAND
      MVC  RB(4),=C'ACC.'           ACCESS ONLY REQUESTED
      CALL ADABAS,(CB,FB,RB)        CALL ADABAS
      CLC  RC,=H'0'                 CHECK RESPONSE CODE
      BNE  EX1ERR                   BRANCH IF NOT 0
```

Issue Find Command

```
      MVC  CCODE,=C'S1'             FIND COMMAND
      MVC  CID,=C'S101'             NONBLANK CID REQUIRED FOR
*                                   IDENTIFICATION OF THE LIST
      MVC  FNR,=H'2'                FILE 2
      MVC  ISNLL,=F'0'              ALL QUALIFYING ISNS DESIRED
      MVC  IBL,=H'0'                ISN BUFFER NOT REQUIRED
      MVI  FB,C'.'                  NO READ OF DATA STORAGE
      MVC  SB(7),=C'XB,3,U.'        SEARCH CRITERION
      MVC  VB(3),=C'099'            SEARCH VALUE
      CALL ADABAS,(CB,FB,RB,SB,VB)  CALL ADABAS
      CLC  RC,=H'0'                 CHECK RESPONSE CODE
      BNE  EX1ERR                   BRANCH IF NOT 0
      CLC  ISNQ,=F'0'               CHECK NUMBER OF ISNS FOUND
      BE   EX1EXIT                  BRANCH TO EXIT IF NO ISNS FOUND
```

Read Each Qualifying Record

```
EX1B  MVC  CCODE,=C'L1'             READ COMMAND
      MVC  ISN,=F'0'                BEGIN WITH 1ST ISN IN LIST
      MVI  COPT2,C'N'               GET NEXT OPTION TO BE USED
      MVC  FB(3),=C'RG.'            ALL FIELDS TO BE RETURNED
EX1C  CALL ADABAS,(CB,FB,RB)        CALL ADABAS
      CLC  RC,=H'0'                 CHECK RESPONSE CODE
      BE   EX1D                     BRANCH IF RESPONSE CODE 0
      CLC  RC,=H'3'                 CHECK IF ALL RECORDS READ
      BE   EX1EXIT                  BRANCH IF YES
      B    EX1ERR                   BRANCH TO ERROR ROUTINE
EX1D  . . .                         PROCESS RECORD . . .
      B    EX1C                     BRANCH TO READ NEXT RECORD
```

Error Routine

```
EX1ERR EQU *
*                                   DISPLAY ERROR MESSAGE
*      .                            TERMINATE USER PROGRAM
```

Issue Close Command

```
EX1EXIT MVC  CCODE,=C'CL'           CLOSE COMMAND
        CALL ADABAS,(CB)            CALL ADABAS
        CLC  RC,=H'0'               CHECK RESPONSE CODE
        BNE  EX1ERR                 BRANCH IF NOT 0
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
  - Field AA = ABCDEFGH
  - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
EXMP2 MVC  CCODE,=C'OP'             OPEN COMMAND
      MVC  RB(6),=C'EXU=1.'         EXCLUSIVE CONTROL REQUESTED
      CALL ADABAS,(CB,FB,RB)        CALL ADABAS
      CLC  RC,=H'0'                 CHECK RESPONSE CODE
      BE   EX2A                     BRANCH IF RESPONSE CODE 0
      B    EX2ERR                   BRANCH IF NOT 0
```

Issue Read Physical Sequential Command

```
EX2A MVC  CID,=C'L201'              NONBLANK CID REQUIRED
     MVC  FNR,=H'1'                 FILE 1 TO BE READ
     MVC  ISN,=F'0'                 ALL RECORDS TO BE READ
     MVC  FB(3),=C'GA.'             VALUES FOR FIELDS AA AND AB
*                                  (GROUP GA) TO BE RETURNED
EX2B MVC  CCODE,=C'L2'              READ PHYS. SEQ.
     CALL ADABAS,(CB,FB,RB)         CALL ADABAS
     CLC  RC,=H'0'                  CHECK RESPONSE CODE
     BE   EX2C                      BRANCH IF RESPONSE CODE 0
     CLC  RC,=H'3'                  CHECK FOR END-OF-FILE
     BE   EX2EXIT                   BRANCH TO EXIT IF END-OF-FILE
     B    EX2ERR                    BRANCH TO ERROR ROUTINE
```

Update Record

- The same fields are to be updated as were read.
- The same CID and format buffer can be used for the update command.
- The ISN of the record to be updated is already in the ISN field as a result of the L2 command.

```
EX2C MVC  CCODE,=C'A1'              UPDATE COMMAND
     MVC  RB(8),=C'ABCDEFGH'        VALUE FOR FIELD AA
     MVC  RB+8(2),=PL2'500'         VALUE FOR FIELD AB
     CALL ADABAS,(CB,FB,RB)         CALL ADABAS
     CLC  RC,=H'0'                  CHECK RESPONSE CODE
     BE   EX2B                      BRANCH TO READ NEXT RECORD
```

Error Routine

```
EX2ERR EQU *
*      .                            DISPLAY ERROR MESSAGE
*      .                            TERMINATE USER PROGRAM
```

Close User Session

```
EX2EXIT MVC  CCODE,=C'CL'            CLOSE COMMAND
        CALL ADABAS,(CB)             CALL ADABAS
        CLC  RC,=H'0'                CHECK RESPONSE CODE
        BNE  EX2ERR                  BRANCH IF NOT 0
        . . .
      ↩
```

Example 3 : User Session with ET Logic

During user session initialization, the user program is to display information indicating the last successfully processed transaction of the previous user session.

For each user transaction, the user program is to

- accept from a terminal 8 characters of input to be used as the key for updating files 1 and 2; and
- issue the Find command for file 1 to determine if a record exists with field AA = input key.

If no record is found, the user program is to issue a message. If a record is found, the user program is to

- delete the record from file 1; and
- add a new record to file 2: Field RA = input key entered.

Other fields are to contain a null value.

If the record cannot be successfully added, the user program is to issue a BT command and display an error message.

If both updates are successful, the user program is to issue an ET command.

- Session Initialization
- Transaction Processing

Session Initialization

The section of the program illustrated is only executed during user session initialization:

Issue Open Command

The OP command is issued with ET data of the previous session being read:

```
EXMP3 EQU  *
      MVC  CCODE,=C'OP'            OPEN COMMAND
      MVI  COPT2,C'E'              ET DATA TO BE READ
      MVC  ADD1,=C'USER0001'       USER IDENTIFICATION
      MVC  ADD3,=C'PASSWORD'       USER PASSWORD
      MVC  RB(8),=C'UPD=1,2.'      FILES 1 AND 2 TO BE UPDATED
      CALL ADABAS,(CB,FB,RB)       CALL ADABAS
      CLC  RC,=H'0'                CHECK RESPONSE CODE
      BE   EX3A                    BRANCH IF RESPONSE CODE 0
      CLC  RC,=H'9'                CHECK FOR RESPONSE CODE 9
      BE   EXMP3                   BRANCH TO REPEAT OPEN
      B    EX3ERR                  BRANCH IF NOT 0 OR 9
EX3A  EQU  *
      CLC  CID,=F'0'               CHECK IF ET DATA FROM
*                                  PREVIOUS SESSION EXISTS
      BE   EX3B                    BRANCH IF NO ET DATA
*      . . .
```

Display ET Data

Display the ET data contained in the record buffer on the terminal screen to inform the user of the last successfully processed transaction of the previous user session:

```
      B    EX3C                    BRANCH TO BEGIN TRANS. PROCESS.
EX3B  EQU  *
```

No ET Data Received

If no ET data was received, a message is displayed indicating that no transactions were successfully processed during the previous user session.

Transaction Processing

This section is executed for each user transaction:

```
EX3C EQU  *
*    . . .                             ACCEPT INPUT FROM TERMINAL  . . .
```

Issue Find Command

Issue the Find command for file 1 to determine if a record exists with the field AA equal to the input key entered:

```
EX3D EQU  *
     MVC  CCODE,=C'S4'                  FIND WITH HOLD COMMAND
     MVC  CID,=C'    '                  ISN LIST NOT TO BE SAVED
     MVC  FNR,=H'1'                     FILE 1
     MVC  ISNLL,=F'0'                   ALL QUALIFY. ISNS TO BE RETURNED
     MVI  FB,C'.'                       NO READ OF DATA STORAGE
     MVC  SB(3),=C'AA.'                 SEARCH CRITERION
     MVC  VB(8),INPUT                   SEARCH VALUE
     CALL ADABAS,(CB,FB,RB,SB,VB,IB)    CALL ADABAS
     CLC  RC,=H'0'                      CHECK RESPONSE CODE
     BE   EX3E                          BRANCH IF RESPONSE CODE 0
     B    EX3ERR                        BRANCH TO ERROR ROUTINE
EX3E EQU  *
     CLC  ISNQ,=F'0'                    CHECK NUMBER OF RECORDS FOUND
     BNE  EX3F                          BRANCH IF RECORD FOUND
```

Issue Message if No Record is Found

If no record is found, the user program issues a message requesting a correction:

```
B    EX3C                          RETURN TO ACCEPT USER INPUT     ↩
```

Delete Record from File 1

The ISN of the record to be deleted is already in the ISN field and in hold status as a result of the S4 command.

```
EX3F EQU  *
     MVC  CCODE,=C'E4'                  DELETE COMMAND
     CALL ADABAS,(CB)                   CALL ADABAS
     CLC  RC,=H'0'                      CHECK RESPONSE CODE
     BE   EX3G                          BRANCH IF RESPONSE CODE 0
     CLC  RC,=H'9'                      CHECK IF CURRENT TRANS. HAS BEEN
*                                       BACKED OUT BY ADABAS
     BE   EX3D                          IF YES, BRANCH TO REPEAT S4
     B    EX3ERR                        BRANCH TO ERROR ROUTINE
```

Add a New Record to File 2

```
EX3G EQU  *
     MVC  CCODE,=C'N1'                 ADD NEW RECORD
     MVC  FNR,=H'2'                    FILE 2
     MVC  FB(6),=C'RA.'                VALUE BEING PROVIDED FOR RA
     MVC  RB(8),INPUT                  VALUE FOR FIELD RA
     CALL ADABAS,(CB,FB,RB)            CALL ADABAS
     CLC  RC,=H'0'                     CHECK RESPONSE CODE
     BE   EX3I                         BRANCH IF RESPONSE CODE 0
     CLC  RC,=H'9'                     WAS TRANSACTION BACKED OUT?
     BE   EX3D                         IF YES, RETURN TO REISSUE TRANS.
```

Unable to Add a New Record

If the attempt to add a new record is not successful, the transaction is backed out and the user is notified that an error condition exists.

```
     MVC  CCODE,=C'BT'                   BACKOUT TRANSACTION
     CALL ADABAS,(CB)                    CALL ADABAS
     CLC  RC,=H'0'                       CHECK IF RESPONSE CODE 0
     BE   EX3H                           BRANCH IF 0
```

Backout Not Successful

When the backout is not successful, a message is issued indicating that result.

```
     B    EX3ERR                         BRANCH TO ERROR ROUTINE
EX3H EQU  *
```

Backout Successful

When the backout is successful, a message is issued indicating that after an error was detected, the transaction was backed out.

```
     B    EX3ERR                         BRANCH TO ERROR ROUTINE
```

Updates Successfully Executed : Issue ET Command with ET Data

When the updates have been successfully executed, an ET command with ET data is issued.

```
EX3I EQU  *
     MVC  CCODE,=C'ET'        END OF TRANSACTION COMMAND
     MVI  COPT2,C'E'          ET DATA TO BE WRITTEN
     MVC  RB(8),INPUT         ET DATA CONSISTS OF INPUT KEY OF THIS TRANSACTION
     CALL ADABAS,(CB,FB,RB)   CALL ADABAS
     CLC  RC,=H'0'            CHECK RESPONSE CODE
     BE   EX3C                IF RESPONSE CODE 0, RETURN TO RECEIVE INPUT FOR
*                             THE NEXT TRANSACTION
     CLC  RC,=H'9'            CHECK IF CURRENT TRANSACTION HAS BEEN BACKED OUT
*                             BY ADABAS
     BE   EX3D                IF CURRENT TRANSACTION HAS BEEN BACKED OUT,
*                             RETURN TO REISSUE TRANSACTION
```

Error Routine

```
EX3ERR EQU  *
*      .                      NONZERO RESPONSE CODE RECEIVED
*      .                      DISPLAY ERROR MESSAGE
*      .                      TERMINATE USER PROGRAM
       . . .
INPUT  DS   CL8               KEY ENTERED FROM TERMINAL
```

ACB COBOL Examples

This section contains examples of using direct Adabas calls in COBOL. The **previously defined Adabas files** are used in each example.

```
*
 *** CONTROL BLOCK
     01  CONTROL-BLOCK.
         02 FILLER                      PIC X(2) VALUE SPACES.
         02 COMMAND-CODE                PIC X(2) VALUE SPACES.
         02 COMMAND-ID                  PIC X(4) VALUE SPACES.
         02 FILE-NUMBER                 PIC S9(4) COMP VALUE +0.
         02 RESPONSE-CODE               PIC S9(4) COMP VALUE +0.
         02 ISN                         PIC S9(8) COMP VALUE +0.
         02 ISN-LOWER-LIMIT             PIC S9(8) COMP VALUE +0.
         02 ISN-QUANTITY                PIC S9(8) COMP VALUE +0.
         02 FORMAT-BUFFER-LENGTH        PIC S9(4) COMP VALUE +100.
         02 RECORD-BUFFER-LENGTH        PIC S9(4) COMP VALUE +250.
         02 SEARCH-BUFFER-LENGTH        PIC S9(4) COMP VALUE +50.
         02 VALUE-BUFFER-LENGTH         PIC S9(4) COMP VALUE +100.
         02 ISN-BUFFER-LENGTH           PIC S9(4) COMP VALUE +20.
         02 COMMAND-OPTION-1            PIC X VALUE SPACE.
         02 COMMAND-OPTION-2            PIC X VALUE SPACE.
         02 ADDITIONS-1                 PIC X(8) VALUE SPACES.
         02 ADDITIONS-2                 PIC X(4) VALUE SPACES.
         02 ADDITIONS-3                 PIC X(8) VALUE SPACES.
         02 ADDITIONS-4                 PIC X(8) VALUE SPACES.
         02 ADDITIONS-5                 PIC X(8) VALUE SPACES.
         02 COMMAND-TIME                PIC S9(8) COMP VALUE +0.
         02 USER-AREA                   PIC X(4) VALUE SPACES.
*
*** USER BUFFER AREAS
    01 FORMAT-BUFFER                    PIC X(100) VALUE SPACES.
    01 RECORD-BUFFER                    PIC X(250) VALUE SPACES.
    01 SEARCH-BUFFER                    PIC X(50)  VALUE SPACES.
    01 VALUE-BUFFER                     PIC X(100) VALUE SPACES.
    01 ISN-BUFFER                       PIC X(20)  VALUE SPACES.
*
```

```
*** ADDITIONAL FIELDS USED IN THE EXAMPLES
    01 PROGRAM-WORK-AREA.
       05         COMM-ID PIC X(4).
       05         COMM-ID-X REDEFINES COMM-ID PIC S9(8) COMP.
       05         INPUT-KEY PIC X(8).
       05         RECORD-BUFFER-EX2.
          10      RECORD-BUFFER-A PIC X(8).
          10      RECORD-BUFFER-B PIC S9(3) COMP-3.
       05         RECORD-BUFFER-EX3.
          10      OPEN-RECORD-BUFFER.
             15   OPEN-RECORD-BUFFER-X PIC X(8).
             15   FILLER PIC S9(8) COMP.
          10      FILLER PIC X(18).
          10      UPDATED-XC PIC X(6).
          10      LAST-XD PIC X(8).
          10      FILLER PIC X(5).
       05         USER-DATA.
          10      RESTART-XD PIC X(8).
          10      RESTART-ISN PIC S9(8) COMP.
       05         SYNC-CHECK-SWITCH PIC 9 VALUE 0.
       05         AB-VALUE PIC S9(4) COMP-3 VALUE +500.
*
```

This section provides the following examples:

- Example 1
- Example 2
- Example 3 : User Session with ET Logic

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
EXMP1.
        MOVE      'OP' TO COMMAND-CODE.
        MOVE      'ACC.' TO RECORD-BUFFER.
        CALL      'ADABAS'
                  USING CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
```

Issue Find Command

```
MOVE      'S1' TO COMMAND-CODE.
        MOVE      'S101' TO COMMAND-ID.
        MOVE      2 TO FILE-NUMBER.
        MOVE      0 TO ISN-LOWER-LIMIT.
        MOVE      0 TO ISN-BUFFER-LENGTH.
        MOVE      '.' TO FORMAT-BUFFER.
        MOVE      'XB,3,U.' TO SEARCH-BUFFER.
        MOVE      '099' TO VALUE-BUFFER.
        CALL      'ADABAS' USING CONTROL-BLOCK, FORMAT-BUFFER,
                  RECORD-BUFFER, SEARCH-BUFFER, VALUE-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
    EX1A.
        IF ISN-QUANTITY = 0  GO TO EX1EXIT.
```

Read Each Qualifying Record

```
EX1B.
        MOVE      'L1' TO COMMAND-CODE.
        MOVE      0 TO ISN.
        MOVE      'N' TO COMMAND-OPTION-2.
        MOVE      'RG.' TO FORMAT-BUFFER.
    EX1C.
        CALL      'ADABAS'
                  USING CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE = 0 GO TO EX1D.
        IF RESPONSE-CODE = 3 GO TO EX1EXIT.
    EX1D.
        . . . PROCESS RECORD . . .
        GO TO EX1C.
```

Error Routine

```
EX1ERR.
*       .DISPLAY ERROR MESSAGE
*       .TERMINATE USER PROGRAM
```

Issue Close Command

```
EX1EXIT.
        MOVE      'CL' TO COMMAND-CODE.
        CALL      'ADABAS' USING CONTROL-BLOCK.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX1ERR.
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
  - Field AA = ABCDEFGH
  - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
EXMP2.
        MOVE      'OP' TO COMMAND-CODE.
        MOVE      'EXU=1.' TO RECORD-BUFFER.
        CALL      'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
```

Issue Read Physical Sequential Command

```
EX2A.
        MOVE      'L201' TO COMMAND-ID.
        MOVE      1 TO FILE-NUMBER.
        MOVE      0 TO ISN.
        MOVE      'GA.' TO FORMAT-BUFFER.
    EX2B.
        MOVE      'L2' TO COMMAND-CODE.
        CALL      'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
        IF RESPONSE-CODE = 0  GO TO EX2C.
        IF RESPONSE-CODE = 3  GO TO EX2EXIT.
        GO TO EX2ERR.
```

Update Record

- The same fields are to be updated as were read.
- The same CID and format buffer can be used for the update command.
- The ISN of the record to be updated is already in the ISN field as a result of the L2 command.

```
EX2C.
        MOVE      'A1' TO COMMAND-CODE.
        MOVE      'ABCDEFGH' TO RECORD-BUFFER-A.
        MOVE      AB-VALUE TO RECORD-BUFFER-B.
        CALL      'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER-EX2.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
        GO TO EX2B.
```

Error Routine

```
EX2ERR.
        . DISPLAY ERROR MESSAGE
        . TERMINATE USER PROGRAM
```

Close User Session

```
EX2EXIT.
        MOVE      'CL' TO COMMAND-CODE.
        CALL      'ADABAS' USING CONTROL-BLOCK.
        IF RESPONSE-CODE NOT EQUAL TO 0  GO TO EX2ERR.
```

Example 3 : User Session with ET Logic

During user session initialization, the user program is to display information indicating the last successfully processed transaction of the previous user session.

For each user transaction, the user program is to

- accept from a terminal 8 characters of input to be used as the key for updating files 1 and 2; and
- issue the Find command for file 1 to determine if a record exists with field AA = input key.

If no record is found, the user program is to issue a message. If a record is found, the user program is to

- delete the record from file 1; and
- add a new record to file 2: Field RA = input key entered.

Other fields are to contain a null value.

If the record cannot be successfully added, the user program is to issue a BT command and display an error message.

If both updates are successful, the user program is to issue an ET command.

Session Initialization

This section of the program is only executed during user session initialization.

- The OP command is issued with ET data of the previous session being read.
- A message is displayed on the terminal screen identifying the last successfully processed transaction of the user's previous session.

```
    EX3.
         MOVE    'OP' TO COMMAND-CODE.
         MOVE    'E' TO COMMAND-OPTION-2.
         MOVE    'USER0002' TO ADDITIONS-1.
         MOVE    'PASSWORD' TO ADDITIONS-3.
         MOVE    'UPD=1,2.' TO RECORD-BUFFER.
         CALL    'ADABAS' USING
                 CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 9  GO TO EX3.
         IF RESPONSE-CODE NOT EQUAL TO 0
              GO TO EX3ERR.
    EX3A.
         MOVE    COMMAND-ID TO COMM-ID.
         IF COMM-ID-X = +0
              GO TO EX3B.
* Display ET data (contained in RECORD BUFFER) on screen to inform user of
* last successfully processed transaction of previous user session.
                . . .DISPLAY ET DATA. . .
              GO TO EX3C.
    EX3B.
*** No ET data received.
*   Display message that no transactions were successfully processed during
*   the previous user session
                . . .DISPLAY MESSAGE . . .
*** Transaction processing.
*   This section is executed for each user transaction.
    EX3C.
*               . . .ACCEPT INPUT FROM TERMINAL. . .
*   Issue Find command for file 1 to determine if record exists with field AA
*   equal to input key entered.
    EX3D.
         MOVE    'S4' TO COMMAND-CODE.
         MOVE    SPACES TO COMMAND-ID.
         MOVE    1 TO FILE-NUMBER.
         MOVE    0 TO ISN-LOWER-LIMIT.
         MOVE    '.' TO FORMAT-BUFFER.
         MOVE    'AA.' TO SEARCH-BUFFER.
         MOVE    INPUT-KEY TO VALUE-BUFFER.
         CALL    'ADABAS' USING
                 CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER,
                 SEARCH-BUFFER, VALUE-BUFFER, ISN-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3E.
         GO TO EX3ERR.
EX3E.
         IF ISN-QUANTITY NOT EQUAL TO ZEROS
              GO TO EX3F.
***No records found, issue message requesting correction.
              . . .ISSUE MESSAGE . . .
         GO TO EX3C.
*** Delete record from file 1.
*  ISN of record to be deleted is already in ISN field and in hold
```

```
status
*  as a result of the S4 command.
    EX3F.
         MOVE     E3' TO COMMAND-CODE.
         CALL     'ADABAS' USING CONTROL-BLOCK.
         IF RESPONSE-CODE = 0
              GO TO EX3G.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
         GO TO EX3ERR.
*** Add new record to file 2.
    EX3G.
         MOVE     'N1' TO COMMAND-CODE.
         MOVE     2 TO FILE-NUMBER.
         MOVE     'RA.' TO FORMAT-BUFFER.
         MOVE     INPUT-KEY TO RECORD-BUFFER.
         CALL     'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3I.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
*** Attempt to add new record not successful.
*  Backout transaction.
*  Notify user that error condition exists.
         MOVE     'BT' TO COMMAND-CODE.
         CALL     'ADABAS' USING control-block.
         IF RESPONSE-CODE = 0
              GO TO EX3H.
*** Backout not successful.
*   Issue message indicating that the backout was not successful
         GO TO EX3ERR.
    EX3H.
*** Backout successful.
*   Issue message indicating the error condition detected while while
adding a
*   new record
         GO TO EX3ERR.
*** Updates successfully executed.
*  Issue ET command with ET data.
  EX3I.
         MOVE     'ET' TO COMMAND-CODE.
         MOVE     'E' TO COMMAND-OPTION-2.
         MOVE     INPUT-KEY TO RECORD-BUFFER.
         CALL     'ADABAS' USING
                  CONTROL-BLOCK, FORMAT-BUFFER, RECORD-BUFFER.
         IF RESPONSE-CODE = 0
              GO TO EX3C.
         IF RESPONSE-CODE = 9
              GO TO EX3D.
*** Error Routine
    EX3ERR.
```

```
*         . DISPLAY ERROR MESSAGE
*         . TERMINATE USER PROGRAM
                 . . .
```

ACB PL/I Examples

This section contains examples of using direct Adabas calls in PL/I . The **previously defined Adabas files** are used in each example.

```
/*** CONTROL BLOCK ***/
DCL 1    CONTROL_BLOCK,
    02   FILLER1                CHAR (2) INIT (' '),
    02   COMMAND_CODE           CHAR (2) INIT (' '),
    02   COMMAND_ID             CHAR (4) INIT (' '),
    02   FILE_NUMBER            BIN FIXED (15) INIT (0),
    02   RESPONSE_CODE          BIN FIXED (15) INIT (0),
    02   ISN                    BIN FIXED (31) INIT (0),
    02   ISN_LOWER_LIMIT        BIN FIXED (31) INIT (0),
    02   ISN_QUANTITY           BIN FIXED (31) INIT (0),
    02   FORMAT_BUFFER_LENGTH   BIN FIXED (15) INIT (100),
    02   RECORD_BUFFER_LENGTH   BIN FIXED (15) INIT (250),
    02   SEARCH_BUFFER_LENGTH   BIN FIXED (15) INIT (50),
    02   VALUE_BUFFER_LENGTH    BIN FIXED (15) INIT (100),
    02   ISN_BUFFER_LENGTH      BIN FIXED (15) INIT (20),
    02   COMMAND_OPTION_1       CHAR(1) INIT (' '),
    02   COMMAND_OPTION_2       CHAR(1) INIT (' '),
    02   ADDITIONS_1            CHAR(8) INIT (' '),
    02   ADDITIONS_2            CHAR(4) INIT (' '),
    02   ADDITIONS_3            CHAR(8) INIT (' '),
    02   ADDITIONS_4            CHAR(8) INIT (' '),
    02   ADDITIONS_5            CHAR(8) INIT (' '),
    02   COMMAND_TIME           BIN FIXED (31) INIT (0),
    02   USER_AREA              CHAR(4) INIT (' ');

/*** USER BUFFER AREAS ***/
DCL FORMAT_BUFFER    CHAR(100),
    RECORD_BUFFER    CHAR(250),
    SEARCH_BUFFER    CHAR(50),
    VALUE_BUFFER     CHAR(100),
    ISN_BUFFER       CHAR(20);
*
*

/***    ADDITIONAL FIELDS USED IN THE EXAMPLES ***/
DCL
    COMM_ID_X   BIN FIXED(31);
    COMM_ID     CHAR(4) BASED (ADDR(COMM_ID_X));
DCL      INPUT_KEY CHAR(8);
DCL      SYNC_CHECK_SWITCH CHAR(1) INIT('0');
DCL 1 RECORD_BUFFER_EX2,
        2   RECORD_BUFFER_A  CHAR(8),
        2   RECORD_BUFFER_B  DEC FIXED(3,0),
        2   FILLER3 CHAR(240);
DCL 1 RECORD_BUFFER_EX3,
        2   OPEN_RECORD_BUFFER,
            3   OPEN_RECORD_BUFFER_X CHAR(8),
            3   FILLER4 BIN FIXED(31),
        2   FILLER5 CHAR(18),
        2   UPDATED_XC CHAR(6),
        2   LAST_XD CHAR(8),
        2   FILLER6 CHAR(5),
    1 USER_DATA,
        2   RESTART_XD CHAR(8),
        2   RESTART_ISN BIN FIXED(31);
DCL     ADABAS ENTRY OPTIONS(ASM);
```

This section provides the following examples:

- Example 1
- Example 2
- Example 3
- PLIADA: Batch/TSO Example

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Issue Open Command

```
/*** Issue Open Command ***/
EXMP1:
    COMMAND_CODE = 'OP';
    RECORD_BUFFER = 'ACC.';
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE > 0        THEN GOTO EX1ERR;
```

Issue Find Command

```
/*** Issue Find Command ***/
    COMMAND_CODE = 'S1';
    COMMAND_ID = 'S101';
    FILE_NUMBER = 2;
    ISN_LOWER_LIMIT = 0;
    ISN_BUFFER_LENGTH = 0;
    FORMAT_BUFFER = '.';
    SEARCH_BUFFER = 'XB,3,U.';
    VALUE_BUFFER = '099';
    CALL ADABAS (CONTROL_BLOCK, FORMAT_BUFFER,
         RECORD_BUFFER, SEARCH_BUFFER, VALUE_BUFFER);
    IF RESPONSE_CODE > 0 THEN GOTO EX1ERR;
EX1A:
    IF ISN_QUANTITY = 0 THEN GOTO EX1EXIT;
EX1B:
    COMMAND_CODE = 'L1';
    ISN = 0;
    COMMAND_OPTION_1 = 'N';
    FORMAT_BUFFER  = 'RG.';
EX1C:
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX1D;
    IF RESPONSE_CODE = 3 THEN
         GOTO EX1EXIT;
EX1D:
    . . .PROCESS RECORD . . .
         GOTO EX1C;
```

Error Routine

```
/*** Error Routine ***/
EX1ERR:
/*   . DISPLAY ERROR MESSAGE */
/*   . TERMINATE USER PROGRAM */
```

Issue Close Command

```
/** Issue Close Command **/
EX1EXIT:
    COMMAND_CODE = 'CL';
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE > 0 THEN
         GOTO EX1ERR;
```

Example 2

- All records in file 1 are to be read in physical sequential order.
- Each record read is to be updated with the following values:
  - Field AA = ABCDEFGH
  - Field AB = 500
- User is to have exclusive control of file 1.

Issue Open Command

```
/*** Issue Open Command ***/
EXMP2:
    COMMAND_CODE = 'OP';
    RECORD_BUFFER = 'EXU=1.';
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
```

Issue Read Physical Sequence Command

```
/*** Issue Read Physical Seq.  Command ***/
EX2A:
    COMMAND_ID = 'L201';
    FILE_NUMBER = 1;
    ISN = 0;
    FORMAT_BUFFER = 'GA.';
EX2B:
    COMMAND_CODE = 'L2';
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN   GOTO EX2C;
    IF RESPONSE_CODE = 3 THEN   GOTO EX2EXIT;
    GOTO EX2ERR;
```

Update Record

```
/*** Update record.  ***/
/* Same fields are to be updated as were read.  */
/* Same CID and FORMAT BUFFER can be used for update.  */
/* ISN of record to be updated is already in ISN field as a result of */
/* the L2 command.  */
EX2C:
    COMMAND_CODE = 'A1';
    RECORD_BUFFER_A = 'ABCDEFGH';
    RECORD_BUFFER_B = 500;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,
                 RECORD_BUFFER_EX2);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
    GOTO EX2B;
```

Error Routine

```
/*** Error Routine ***/
EX2ERR:
/*   . DISPLAY ERROR MESSAGE */
/*   . TERMINATE USER PROGRAM */
```

Close User Session

```
/* Close User Session */
EX2EXIT:
    COMMAND_CODE = 'CL';
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE > 0 THEN   GOTO EX2ERR;
```

Example 3

This example illustrates a user session with ET logic. The user program is to perform the following functions:

1. During user session initialization, display information indicating the last successfully processed transaction of the previous user session.
2. For each user transaction:

- Accept from a terminal 8 characters of input that is used as the key for updating files 1 and 2.
- Issue a Find command for file 1 to determine if a record exists with field AA = input key.
- If no record is found, issue a message.
- If a record is found:
  - Delete the record from file 1;
  - Add a new record to file 2: Field RA = input key entered. Other fields to contain null value.
  - If the record cannot be successfully added, issue a BT command, display error message.
  - If both updates are successful, issue an ET command.

Session Initialization

This section of the program is only executed during user session initialization.

- The OP command is issued with ET data of the previous session being read.
- A message is displayed on the terminal screen identifying the last successfully processed transaction of the user's previous session.

```
EX3:
    COMMAND_CODE = 'OP';
    COMMAND_OPTION_2 = 'E';
    ADDITIONS_1 = 'USER0003';
    ADDITIONS_3 = 'PASSWORD';
    RECORD_BUFFER = 'UPD=1,2.';
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 9 THEN   GOTO EX3;
    IF RESPONSE_CODE > 0 THEN
         GOTO EX3ERR;
```

```
EX3A:
    COMM_ID = COMMAND_ID;
    IF COMM_ID_X = 0 THEN
         GOTO EX3B;
/*  Display ET data (contained in RECORD BUFFER) on screen to inform user of
    last successfully processed transaction of previous user session. */
         . . .DISPLAY ET DATA. . .
    GOTO EX3C;
EX3B:
/*                          */
/*** No ET data received.  */
/*   Display message that no transactions were successfully processed during
    the previous user session. */
         . . .DISPLAY MESSAGE . . .
/*                               */
/*** Transaction processing.  ***/
/* This section is executed for each user transaction.  */
EX3C:
         . . .ACCEPT INPUT FROM TERMINAL. . .
/*                                                        */
/* Issue Find command for file 1 to determine if rec exists with field AA
   equal to input key entered. */
EX3D:
    COMMAND_CODE = 'S4';
    COMMAND_ID = ' ';
    FILE_NUMBER = 1;
    ISN_LOWER_LIMIT = 0;
    FORMAT_BUFFER = '.';
    SEARCH_BUFFER = 'AA.';
    VALUE_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER,
         SEARCH_BUFFER,VALUE_BUFFER,ISN_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3E;
    GOTO EX3ERR;
EX3E:
    IF ISN_QUANTITY > 0 THEN
         GOTO EX3F;
/*                                                        */
/* No record found, issue message requesting correction.  */
         . . .ISSUE MESSAGE . . .
    GOTO EX3C;
/*                              */
/* Delete record from file 1.  */
/* ISN of record to be deleted is already in ISN field and in hold
status
   as a result of the S4 command. */
EX3F:
    COMMAND_CODE = 'E4';
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3G;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
    GOTO EX3ERR;
/***Add new record to file 2.  */
EX3G:
    COMMAND_CODE = 'N1';
    FILE_NUMBER = 2;
    FORMAT_BUFFER = 'RA.';
    RECORD_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3I;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
/*                                */
/*  Attempt to add new record not successful. Backout transaction and
notify
    user that error condition exists. */
    COMMAND_CODE = 'BT';
    CALL ADABAS (CONTROL_BLOCK);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3H;
/*                            */
/*  Backout not successful.   */
     . . .ISSUE MESSAGE INDICATING BACKOUT NOT SUCCESSFUL . .
    GO TO EX3ERR.
/*                               */
EX3H:
/*** Backout successful.    ***/
/* Issue message indicating error condition detected while adding new
record.*/
         . . .ISSUE MESSAGE. . .
    GOTO EX3ERR;
/*                                     */
/*** Updates successfully executed.  ***/
/*   Issue ET command with ET data.    */
EX3I:
    COMMAND_CODE = 'ET';
    COMMAND_OPTION_2 = 'E';
    RECORD_BUFFER = INPUT_KEY;
    CALL ADABAS (CONTROL_BLOCK,FORMAT_BUFFER,RECORD_BUFFER);
    IF RESPONSE_CODE = 0 THEN
         GOTO EX3C;
    IF RESPONSE_CODE = 9 THEN
         GOTO EX3D;
/*                                    */
/***    Error Routine    ***/
EX3ERR:
/*  . DISPLAY ERROR MESSAGE */
/*  . TERMINATE USER PROGRAM */
         . . .
```

PLIADA: Batch/TSO Example

PLIADA is a PL/I program you could run in a batch/TSO environment. Its source can be found in the PLIADA member of the ADAvrs.MVSSRCE library. It is written to conform to the language specification of Enterprise PL/I for z/OS V4.1.

This program will issue L1 commands to list from the sample Employees file provided with Adabas. Refer to member ADALODE of the MVSJOBS dataset to load the file.

PLIXADA should be link-edited with the ADAUSER load module, in which case the CALL statement should use the symbol ADABAS as the program being invoked.

The following table lists the PL/I sample components for ACB programs:

Member	Description
PLIADA	PL/I Sample ADACB main program
PADABAS	PL/I ADACB structure declaration
PADACALL	PL/I ADABAS entry with APL linkage declarations

The IBM cataloged procedures IBMZC, IBMZCB or IBMZCBG may be used to compile, compile and bind (link) or compile, bind (link) and execute the PLIADAX program. Consult the appropriate IBM documentation for information on compiling, linking and executing an Enterprise PL/I module. Your installation may have standards for these procedures or different JCL procedures for compiling, linking and executing an Enterprise PL/I module.

ACB Fortran Example

This section contains an example of using direct Adabas calls in FORTRAN. The **previously defined Adabas files** are used in each example.

```
C   *** CONTROL BLOCK ***
    INTEGER*4     CB(20),CID,ISN,ISNL,ISNQ
    INTEGER*4     ADD1(2),ADD2,ADD3(2),ADD4(2),ADD5(2)
    INTEGER*4     CTIME,UAREA
    INTEGER*2     CBI(40),CCODE,FNR,RC,FBL,RBL,SBL,VBL,IBL
    LOGICAL*1     CBL(80),COPT1,COPT2
    EQUIVALENCE       (CB(1),CBI(1),CBL(1))
    EQUIVALENCE       (CID,CB(2)),(ISN,CB(4))
    EQUIVALENCE       (ISNL,CB(5)),(ISNQ,CB(6))
    EQUIVALENCE       (ADD1(1),CB(10)),(ADD2,CB(12)),(ADD3(1),CB(13))
    EQUIVALENCE       (ADD4(1),CB(15),(ADD5(1),CB(17))
    EQUIVALENCE       (CTIME,CB(19)),(UAREA,CB(20))
    EQUIVALENCE       (CCODE,CBI(2)),(FNR,CBI(5)),(RC,CBI(6))
    EQUIVALENCE       (FBL,CBI(13)),(RBL,CBI(14)),(SBL,CBI(15))
    EQUIVALENCE       (VBL,CBI(16)),(IBL,CBI(17))
    EQUIVALENCE       (COPT1,CBL(35)),(COPT2,CBL(36))


C   *** USER BUFFER AREAS ***
    INTEGER*4 FB(25),RB(50),SB(10),VB(10),IB(50)
 *
 *
C   *** ADDITIONAL FIELDS USED IN THIS EXAMPLE  ***
    LOGICAL*1 BLANK/1H /,COPH/1HH/,PERIOD/1H./,COPN/1HN/
    INTEGER*2 S1/2HS1/,L1/2HL1/,CL/2HCL/
    INTEGER*4 CID1/4HS101/,FB1/4H.  /,FB2/4HRG. /,SB1/4HXB,3/
INTEGER*4 SB2/4H,U. /,VB1/4H099 /
```

This section provides the following example:

- Example 1

Example 1

- Find the set of records in file 2 with XB = 99.
- Read each record selected using the GET NEXT option.

Initialize Control Block

```
c*** Initialize Control Block
    DO 5 I=1,80
    CBL(I)=BLANK
5   CONTINUE
    DO 10 I=3,6
    CB(I)=0
10  CONTINUE
    CBI(13)=100
    CBI(14)=200
    CBI(15)=40
    CBI(16)=40
    CBI(17)=200
    CBI(19)=0
```

Issue Find Command

```
c***Issue FIND Command
    CCODE=S1
    CID=CID1
    FNR=2
    ISNL=0
    COPT1=COPH
    FB(1)=FB1
    SB(1)=SB1
    SB(2)=SB2
    VB(1)=VB1
    CALL ADABAS(CB,FB,RB,SB,VB,IB)
    IF(RC.NE.0) GO TO 50
    IF(ISNQ.EQ.0) GO TO 100
```

Read Each Record Selected

```
c***Read Each Record Selected
15  CONTINUE
    CCODE=L1
    ISN=0
    COPT1=COPN
    FB(1)=FB2
    CALL ADABAS(CB,FB,RB)
    IF(RC.EQ.0) GO TO 30
    IF(RC.EQ.3) GO TO 100
    PRINT 60,RC,CCODE
60  FORMAT(1H0,'ADABAS ERROR CODE',I4,' FROM '.A2,' COMMAND')
    GO TO 50
30  CONTINUE
C   ...PROCESS RECORD...
    GO TO 15
50  CONTINUE
    STOP
100 CONTINUE
    ...
```

46 ACBX Examples

Two sample COBOL programs are provided to illustrate how to make Adabas ACBX direct calls:

- COBADA8 is a COBOL program you could run in a batch/TSO environment. Its source can be found in the COBADA8 member of the ADA*vrs*.MVSSRCE library.
- COBACI8 is a COBOL program you could run under CICS/TS. Its source can be found in the COBACI8 member of the ACI*vrs*.MVSSRCE library.

The source modules for both programs should be compiled using the Enterprise COBOL compiler Version 3 Release 3 or later.

Note: These programs are provided "as is" and will not be supported by Software AG.

This chapter describes examples of direct calls using the ACBX interface. The last two examples are sample PL/I programs to illustrate ACBX calls.

COBADA8: Batch/TSO Example

COBADA8 is a COBOL program you could run in a batch/TSO environment. Its source can be found in the COBADA8 member of the ADA*vrs*.MVSSRCE library.

Before compiling COBADA8, review the source, especially the comments so the proper database ID and file number are used in the WORK-DBID and WORK-FNR fields defined in the WORKING-STORAGE section of the program.

COBADA8 should be link-edited with the ADAUSER load module in which case the CALL statement should use the literal "ADABAS" as the program being invoked.

If COBADA8 is compiled and link-edited as a reentrant module, use the LNK-NAME field in the CALL statement and supply a value of "ADAUSER" on the LINK-NAME definition in WORKING-STORAGE.

The IBM cataloged procedures IGYWC, IGYWCL or IGYWCLG may be used to compile, compile and link or compile, link and execute the COBADA8 program. Consult the appropriate IBM documentation for information on compiling, linking and executing an Enterprise COBOL module. Your installation may have standards for these procedures or different JCL procedures for compiling, linking and executing an Enterprise COBOL module.

COBACI8: CICS/TS Example

COBACI8 is a COBOL program you could run under CICS/TS. Its source can be found in the COBACI8 member of the ACI*vrs*.MVSSRCE library.

Before compiling COBACI8, review the source member and provide the proper values for WORK-DBID and WORK-FNR in the WORKING-STORAGE section. In addition, supply the proper value for the LINK-NAME field to name the Adabas Version 8 CICS application stub that should be the object of the EXEC CICS LINK command used to invoke Adabas CICS services. This name should match the name provided in the Adabas CICS globals table, keyword ENTPT, used in the CICS region where COBACI8 is to be executed.

The COBACI8 program is intended to be compiled with the built-in CICS pre-processor support provided by the Enterprise COBOL compiler. After compiling and linking the source module with the appropriate cataloged procedure or JCL, the module COBACI8 and a transaction to execute it must be defined to CICS using RDO or the CEDA facility. The COBACI8 module uses the CICS COMMAREA to communicate with the Adabas Version 8 CICS application stub. It is not necessary to define a CICS TWA (Transaction Work Area) to execute the COBACI8 with a CICS transaction.

The following table lists the PL/I sample components for ACBX programs under CICS:

Member	Description
PLIADAC	PL/I Sample ADACBX main program
PADABASX	PL/I ADACBX and ADABDX structure declarations
PDACOMX	PL/I CICS CommArea for ADACBX

The IBM cataloged procedures IBMZC, IBMZCB or IBMZCBG may be used to compile, compile and bind (link) or compile, bind (link) and execute the PLIADAX program. Consult the appropriate IBM documentation for information on compiling, linking and executing an Enterprise PL/I module. Your installation may have standards for these procedures or different JCL procedures for compiling, linking and executing an Enterprise PL/I module.

PLIADAX: Batch/TSO Example

PLIADAX is a PL/I program you could run in a batch/TSO environment. Its source can be found in the PLIADAX member of the ADAvrs.MVSSRCE library. It is written to conform to the language specification of Enterprise PL/I for z/OS V4.1.

This program will issue L1 commands to list from the sample Employees file provided with Adabas. Refer to member ADALODE of the MVSJOBS dataset to load the file.

Before compiling PLIADAX review the source, especially the comments, so the proper file number is used.

PLIXADAX should be link-edited with the ADAUSER load module, in which case the CALL statement should use the symbol ADABAS as the program being invoked.

The following table lists the PL/I sample components for ACBX programs:

Member	Description
PLIADAX	PL/I Sample ADACBX main program
PADABASX	PL/I ADACBX and ADABDX structure declarations
PADACALX	PL/I ADABAS entry with APLX linkage declarations

The IBM cataloged procedures IBMZC, IBMZCB or IBMZCBG may be used to compile, compile and bind (link) or compile, bind (link) and execute the PLIADAX program. Consult the appropriate IBM documentation for information on compiling, linking and executing an Enterprise PL/I module. Your installation may have standards for these procedures or different JCL procedures for compiling, linking and executing an Enterprise PL/I module.

PLIADAC: CICS/TS Example

PLIADAC is a PL/I program you could run under CICS/TS. Its source can be found in the PLIADAC member of the ACIvrs.MVSSRCE library. It is written to conform to the language specification of Enterprise PL/I for z/OS V4.1.

This program will issue L1 commands to list from the sample Employees file provided with Adabas. Refer to member ADALODE of the MVSJOBS dataset to load the file.

Before compiling PLIADAC review the source, especially the comments, so the proper file number is used. In addition, supply the proper value for the variable ADALnkName to specify the name of the Adabas Version 8 CICS application stub that should be the object of the EXEC CICS LINK command used to invoke Adabas CICS services. This name should match the name provided in the Adabas CICS globals table, keyword ENTPT, used in the CICS region where PLIADAC is to be executed.

The PLIADAC program is intended to be compiled with the built-in CICS pre-processor support provided by the Enterprise PL/I compiler. After compiling and linking/binding the source module using a appropriate cataloged procedure or JCL, the module PLIADAC and a transaction to execute it must be defined to CICS using RDO or the CEDA facility. The PLIADAC module uses the CICS COMMAREA to communicate with the Adabas Version 8 CICS application stub. It is not necessary to define a CICS TWA (Transaction Work Area) to execute PLIADAC with a CICS transaction.

Member	Description
PLIADAC	PL/I Sample ADACBX main program
PADABASX	PL/I ADACBX and ADABDX structure declarations
PDACOMX	PL/I CICS CommArea for ADACBX

The IBM cataloged procedures IBMZC, IBMZCB or IBMZCBG may be used to compile, compile and bind (link) or compile, bind (link) and execute the PLIADAX program. Consult the appropriate IBM documentation for information on compiling, linking and executing an Enterprise PL/I module. Your installation may have standards for these procedures or different JCL procedures for compiling, linking and executing an Enterprise PL/I module.

Index