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*. 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.

This chapter covers the following topics:

- Adabas Control Block (ACB)
- Extended Adabas Control Block (ACBX)
- Differences between the ACB and the ACBX
- Logging the Control Blocks

Adabas Control Block (ACB)

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

- ACB Format
- ACB Fields
- ACB DSECT
- 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

DSECT Name	Field	Control Block Position	Offset	Length (in Bytes)	Format
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*.

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*.

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. For OP commands, an optional user-specific non-activity timeout value can be specified in this field. An OP command also returns Adabas release information in this field (see also the Additions 5 field description). When using the multifetch option, this field holds an optional maximum count of prefetched records to return; if zero, there is no limit.

ISN Quantity (ACBISQ)

The ISN quantity 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. The OP command uses this field to specify an optional user-specific transaction time limit; it returns system and call type information flags in the ISN quantity field (see also the Additions 5 field description). 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 addition 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*.

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 Format
- ACBX Fields
- 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

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
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
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

DSECT Field Name	Field	Control Block Position	Offset	Length (in bytes)	Format
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*.

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.

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* 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)

The ISN Lower Limit field specifies the starting point in an ISN list or range where processing is to begin.

For OP commands, an optional user-specific non-activity timeout value can be specified in this field. The OP command also returns Adabas release information in this field.

When the multifetch option is used, this field holds an optional maximum count of records to be prefetched; if zero, there is no limit.

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 the 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 segment from a larger pool of ISNs by this operation. Search commands using security-by-value set the ISN Quantity to "1" when *more than one* ISN meets the search criteria.

For an OP command, an optional user-specific transaction time limit may be specified in this field. The OP command returns system and call type information in this field.

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 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.

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*.

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.
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.

ACBX DSECT Name	Description
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)

Offset	ACB DSECT Field Name	ACBX DSECT Field Name
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)

Offset	ACB DSECT Field Name	ACBX DSECT Field Name
54	---	ACBXADD5 (Additions 5)
5C	---	ACBXADD6 (Additions 6)
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

There are two formats for the Command log (CLOG), CLOGLAYOUT=5 and CLOGLAYOUT=8. The ADARUN parameter CLOGLAYOUT determines which format is used.

Note:
We recommend that you use CLOGLAYOUT=5 for user programs designed for versions of Adabas up to Adabas 8. User programs designed for Adabas 8 or later should use CLOGLAYOUT=8.

If CLOGLAYOUT=5, all Adabas control block fields are logged in the basic command logging area. If CLOGLAYOUT=8, each of the buffers are written out in much the same way as for CLOGLAYOUT=5, except that each buffer is prefixed by its corresponding Adabas buffer description (ABD). Each segmented buffer (format, record, or multifetch) is written separately and uniquely identified.

The CLOGLAYOUT parameter is described in *CLOGLAYOUT : Command Logging Format* . A detailed description of both command log formats can be found in *Command Log Formats*.