User Exit 4 (User-Generated Log Data)

User exit 4 is called immediately before an Adabas command log record is to be written. It may be used to generate any required user log data (SMF records) special statistics, or to suppress writing a log record.

Note:
User exit 4 is still called even if ADARUN LOGGING=NO. The only way to disable user exit 4 is to remove the ADARUN UEX4 parameter from the Adabas run.

This chapter covers the following topics:

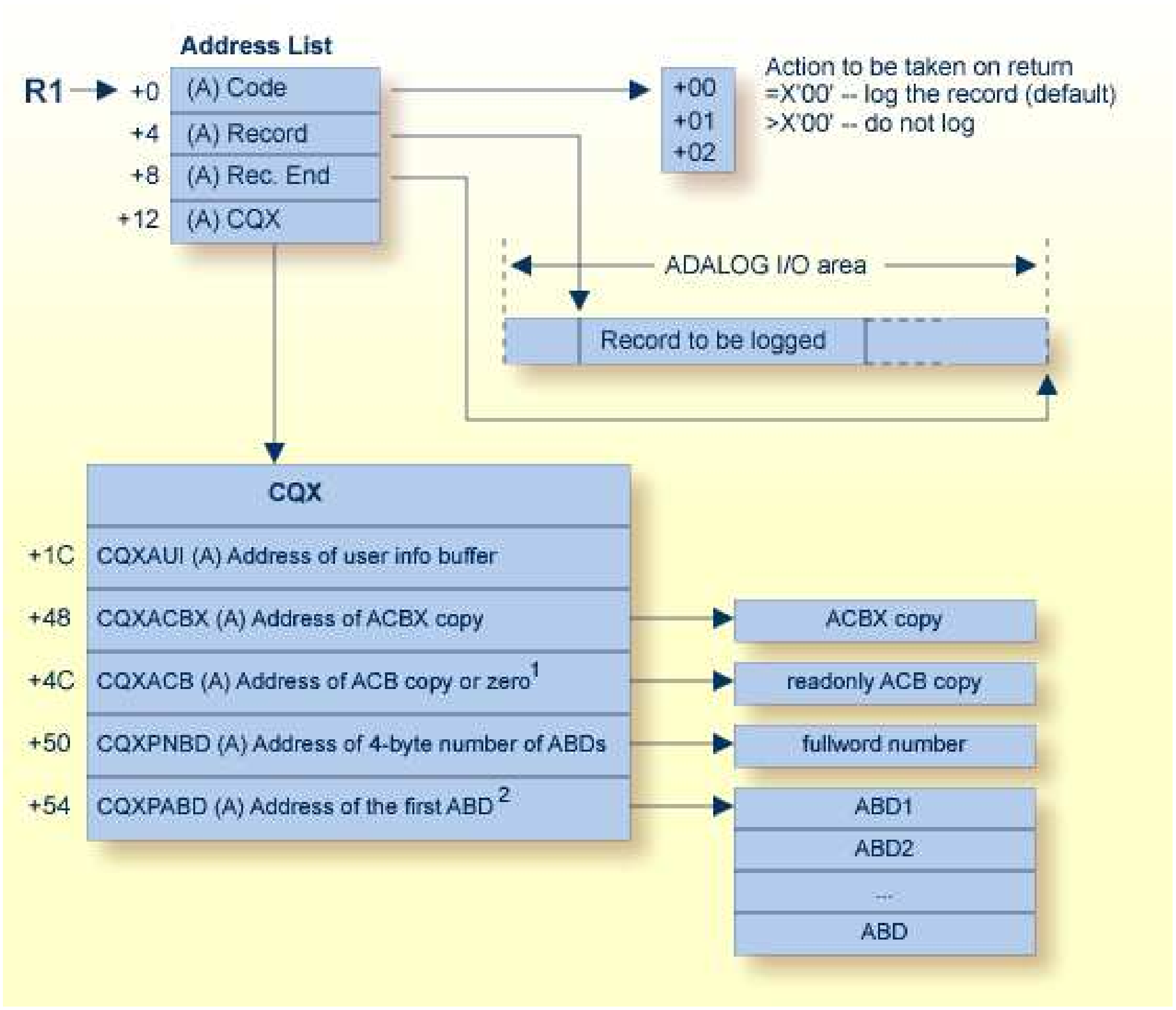
Command Log Format

Adabas supports two different command log formats. The ADARUN CLOGLAYOUT parameter determines which format is used:

- CLOGLAYOUT=5 (the default) is supported only in Adabas versions 5.2 and above.
- CLOGLAYOUT=8 specifies the new format, which is supported only in Adabas versions 8 and above.

Both formats are described in *Command Log Formats*.

Ensure that your user exit and command log evaluation programs recognize the format in use before switching to it.


Address List
R1
+0
(A) Code
+4
(A) Record
+8
(A) Rec. End
+12
(A) CQX
+00
+01
+02
Action to be taken on return
=X'00' -- log the record (default)
>X'00' -- do not log
ADALOG I/O area
Record to be logged
CQX
+1C CQXAUI (A) Address of user info buffer
+48 CQXACBX (A) Address of ACBX copy
+4C CQXACB (A) Address of ACB copy or zero1
+50 CQXPNBD (A) Address of 4-byte number of ABDs
+54 CQXPABD (A) Address of the first ABD 2
ACBX copy
readonly ACB copy
fullword number
ABD1
ABD2
...
ABD

User-Generated Log Data User Exit (4) Parameters

[1] *Address of ACB Copy*: This address should be set to zero if the command is using an ACBX interface direct call.

[2] *Address of the first ABD*: The Adabas buffer descriptions (ABDs) are in a contiguous array. For complete information about locating ABDs in this array, read *Locating the Correct ABD*, elsewhere in this section.

Parameter	Address of . . .
0(R1)	a byte containing a logging action code. This byte contains: ● +00 -- action code to log the record upon each call. If changed to a nonzero value, this record will not be written to DDLOG. ● +01 -- reserved for future use ● +02 -- two-byte database ID.
4(R1)	the record to be logged. This address is zero if the exit is called at the end of the nucleus session.
8(R1)	the end of the Adabas I/O area. This address is zero if the exit is called at the end of the nucleus session.
12(R1)	the command queue element (CQX). This address is zero if the exit is called at the end of the nucleus session.

The record to be logged may be modified by the user exit. The record’s address in 4(R1) may also be modified. The logging action code must always be specified before returning to the Adabas nucleus.

Warning:
When modifying the record, do not exceed the end address of the ADALOG I/O area contained in 8(R1).

Locating the Correct ABD

Internally, Adabas 8 only uses extended Adabas control blocks (ACBX) and Adabas buffer descriptions (ABDs). Direct calls made using the classic Adabas control block (ACB) and buffer definitions have their data structures converted to ACBX calls and ABDs by ADASVC before the nucleus sees the call. Thus, the protocol for locating and accessing buffers in user exits, such as this one, has changed as of Adabas 8.

The Adabas buffer descriptions (ABDs) are now in a contiguous array. However, the internal representation of the ABD may not have the same length as the base ABD, as defined by the value of the ABDXQLL symbol in the ADABDX DSECT, although the first ABDXQLL bytes continue to be mapped by ADABDX. This means that you should not use the ABDXQLL value in the ADABDX DSECT to locate the next ABD in the ABD array. Instead, you should use the value of the two-byte ABDXLEN field at offset `+X'00'` of the ABD to determine the end of that ABD and the start of the next ABD in the array. Do not assume that all internal ABD representations have the same length: each must be located in turn by applying its predecessor’s ABDXLEN value.

In addition, the order of the ABDs is not defined and my change over time or from command to command, although within the array all ABDs of a given type (format buffer, record buffer, etc.) are contiguous. There will be an ABD for every buffer provided by the user that is documented as an input or output buffer for the specific command. There may also be additional buffers created by other components. When there are multiple instances of format, record and (optional) multifetch buffers, they are related based on their position: the first format buffer is associated with the first record (and optional multifetch) buffer, the second with the second, and so forth. If the caller provides an unequal number of format, record and (optional) multifetch buffers, dummy descriptors with a zero buffer length are created to bring about equal quantities. When multifetch is used with a classic ACB call, certain commands (L1/2/3/4/9) will have their ISN buffer converted into a multifetch buffer. Here are some examples:

- If a caller (using either an ACB or ACBX call) issues an OP command and provides a record buffer and search buffer, the array of ABDs will have one record buffer ABD and one dummy format buffer ABD (to satisfy the internal requirement that there be equal numbers of format and record buffers). There is no ABD for the search buffer because that is not a documented input or output buffer for the OP command.

- If a caller uses an ACBX call to issue an L1 command and provides two format buffers and three record buffers, the array of ABDs will have three record ABDs and three format ABDs, the last one of which is a dummy format ABD. The first record buffer is associated with the first format buffer; the second record buffer is associated with the second format buffer; and the third record buffer is associated with the third (dummy) format buffer.

- Suppose a caller uses an ACB call to issue an L3 command with Command Option 1 set to "M" (multifetch) and Command Option 2 set to "A" (ascending retrieval from a specified value). In addition, the caller provides a format buffer, a record buffer, an ISN buffer, a search buffer and a value buffer. In this case, the array of ABDs will have one format buffer ABD, one record buffer ABD, one multifetch buffer ABD, one search buffer ABD, and one value buffer ABD. The caller’s ISN buffer will have been converted to a multifetch buffer.