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)*. 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
- Using the User Buffer with ADALNK User Exits 1 and 2
- Using the User Buffer with Adabas Nucleus Exits 4 and 11

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*.

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 when the target completes the command. 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 UBLUINFO has the physical length of the buffer. The user buffer itself is found immediately after the user block at label UBUINFO. The first two bytes of the user buffer contain the length of the buffer including the length field. In ADALNK user exit 1, you can decrease the effective size of the buffer in UBLUINFO for this one command. 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...’

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)*.

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
- Example

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* 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...’
ACBX	x’30’	x’0030 0000...’