User Exit 11 (General Processing)

This user exit is given control by Adabas immediately after a command is received by the Adabas nucleus. The command itself has yet to be processed except for the determination of the type of command (simple access, complex access, update).

One of the most common applications of this user exit is to insert a security password or a cipher code into the ACBX.

This user exit functionality largely matches that of the classic user exit 1, except for the fact that edited copies of the CQX and ACBX data structures are used during user exit 11 processing, rather than the actual structures used by user exit 1. In addition, support for user exit 1 is dropped in Adabas 8 (or later).

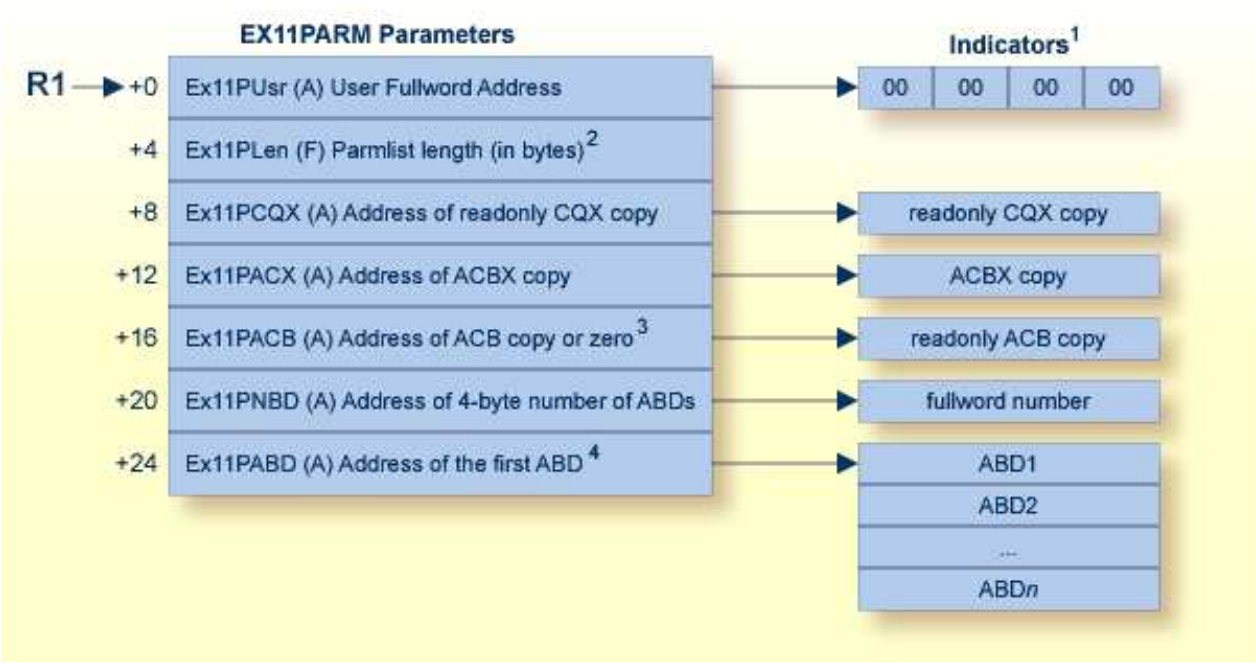
Only certain fields in the ACBX may be changed by the exit: ACBXFNR (file number), ACBXADD2 (Additions 2), ACBXADD3 (Additions 3), and ACBXUSER (user area). The nucleus will ignore changes in any other ACBX fields and all changes to the CQX. DSECT EX11PARM maps the user exit 11 parameter list. In addition, a sample user exit 11 skeleton called UEX11 is provided. Both the DSECT and the user exit skeleton are provided in the Adabas source library.

The call to the user exit is made using a standard BASR 14,15 assembler instruction. Register 1 contains the address of a parameter list. All registers must be saved when control is received and restored immediately prior to returning control to Adabas, with the exception of Register 15 which contains the return code. A non-zero value means that the command should not be executed and returns response code 22.

Notes:

1. All user exits must return the same program status word (PSW) fields to the calling program that were active when the user exit was called. This applies in particular to the addressing mode (AMODE), program mask, problem state flag, PSW key, and address space control setting. The condition code need not be preserved. If any of these PSW fields is changed by the user exit, one way to ensure that their previous values are returned is to envelope the code where the change is in effect with a pair of the BAKR ... PR instructions.
2. The file number specified in the original ACBX for the call cannot be changed. If it is necessary to change a file number with the user exit, change the field ACBXFNR field in the copy of the ACBX that is made for the exit.
3. The command code field in the Adabas control block cannot be changed; a response code is returned if you attempt to do so.
4. The length of an Adabas buffer in any Adabas buffer description (ABD) used by the call cannot be changed.

Input and Output Parameters



General Processing User Exit (1) Parameters

[1] *Indicators*: Before calling user exit 11, the fullword indicator areas are set to zero.

[2] *Parmlist length*: The EX11PARM parameter list length is at least 28 bytes.

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

[4] *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*, next in this section.

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.