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*.

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.

This chapter covers the following topics:

- How Adabas Distinguishes Between ACB and ACBX Direct Calls
- Specifying an ACB Interface Direct Call
- Specifying an ACBX Interface Direct Call
- Mixing ACB and ACBX Direct Calls

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*.

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*. For information about the relationships between different ABD types, read *Understanding the Different Buffer Types*.

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

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.
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*.

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*.	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*.