Format Buffers

Format buffers specify the fields to be processed during the execution of an Adabas read or update command. The format buffer must be long enough to hold the largest field definition contained in the related program, including the ending point (.) of the buffer itself.

Format buffers are used in combination with record buffers and, in ACBX interface direct calls, multifetch buffers. If a format buffer is specified, a record buffer must also be specified. If a record buffer is not provided, Adabas will create dummy record buffer (with length zero) to pair with the format buffer.

When using the ACBX direct call interface, multiple format buffers can be specified for an Adabas direct call.

In an ACBX interface direct calls, a multifetch buffer segment may also be listed. For more information, read *Record Buffers* and *Multifetch Buffers*.

For complete information about the relationships between the different types of ABD or buffer specifications, read *Understanding the Different Buffer Types*.

This chapter covers the following topics:

- Format Buffer Syntax
- Field Selection Criteria
- Record Format Specifications
- Specifying Field Lengths of LA (Long Alpha) Fields in Format Buffers
- Format Buffer Performance Considerations

Format Buffer Syntax

Format buffers are divided into two parts: optional field selection criteria for the buffer and the record format expected.

Multiple record formats may be specified in the format buffer for files that contain various record formats. The specific record format to be used is determined by the value of a field in the file (for example, record type). When specifying multiple record formats, each one must be preceded by a format selection criterion.

The format buffer has the following syntax:

```
[field-selection-criteria1] record-format1[,[field-selection-criteria2] record-format2]... .
```

A comma must be used to separate all format buffer entries. One or more spaces may be present between entries. The last entry may not be followed by a comma.

The format buffer must end with a period.

The following sections describe the components of this syntax:

- *Field Selection Criteria*
- *Record Format Specifications*

Field Selection Criteria

Field selection criteria (*field-selection-criteria1* and *field-selection-criteria2*) are optional in a format buffer. They allow you to restrict record formats to specific values of fields. The syntax of field selection criteria is:

```
(field-name operator value1 [, value2]...)
```

field-name

The field name used in field selection criteria must be the valid name of a field in the FDT of the Adabas file being read. It *cannot* be:

- The name of a group or periodic group
- A field using any of the MU, PE, LA, or LB options
- A subfield, superfield, subdescriptor, or superdescriptor
- A collation descriptor
- A hyperdescriptor
- A phonetic descriptor.

In addition, fields specified with the NU or NC/NN options must have a non-null value; otherwise the selection criteria will be false.

operator

The following table lists the operators that can be used in the format buffer field selection criteria.

Operator	Meaning
EQ	equal to
=	equal to
NE	not equal to
LT	less than
<	less than
GT	greater than
>	greater than
LE	less than or equal to
GE	greater than or equal to

value1, value2

The value must be a numeric integer or an alphanumeric value.

If you use the EQ or = operator, a series of values can be specified, separated by commas. An alphanumeric value must be enclosed within apostrophes (for example, `'value'`).

Consider the following example:

```
(SA = 1) record-format-1, (SA = 2,3,4) record-format-2, (SA GE 4) record-format-3.
```

The field selection criteria specifies that:

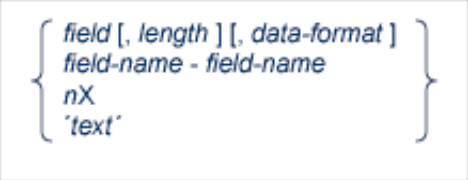
- If the value of field SA is "1", record-format-1 is used;
- If the value of field SA is "2", "3" or "4", record-format-2 is used;
- If the value of field SA is equal to or greater than "4", record-format-3 is used.

The first criterion that is met is used. If no criteria are met, a response code is returned. In the example above, if the value of field SA is "4", both the last two conditions in the field selection criteria are satisfied. However, record-format-2 will always be used, rather than record-format 3 because it is the first criteria satisfied. Likewise, if the value of SA is "0", a response code is returned.

Record Format Specifications

The record format (*record-format1*) is required and is used to indicate which fields should be read or updated in the Adabas direct call. Additional record formats can also be specified (*record-format2*).

The syntax of the record format is:

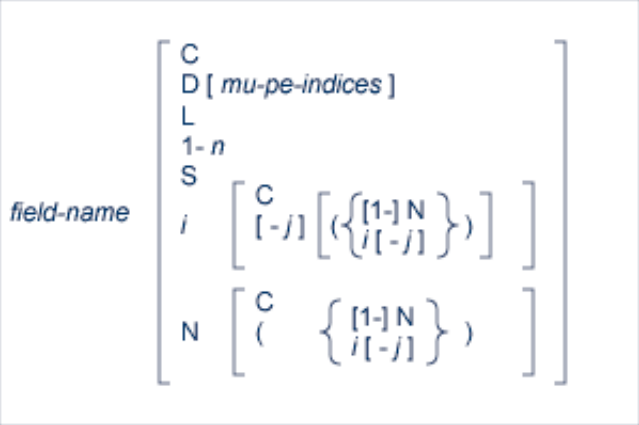


For information about each of the elements in this syntax, read the section listed in the following table:

Syntax Element	Read
field	*field Syntax*
length	*Length and Data Format*
data-format	*Length and Data Format*
field-name - field-name	*Field Series Notation*
*n*X	*Space Notation (nX)*
'text'	*Text Insertion Notation*

field Syntax

The syntax for a *field* in record format syntax is:



Field Syntax

This section covers the following topics related to this field syntax:

- field-name Specifications
- Index or Range Notation (i [-j] Notation)
- Periodic Group References
- Multiple-Value Fields
- Multiple-Value Fields within Periodic Groups
- Count Indicator (C)
- Daylight Savings Indicator (D)
- Length Indicator (L)
- Highest Occurrence/Value Indicator (N)
- SQL Significance Indicator (S)

- Selecting LOB Values or LOB Value Segments

field-name Specifications

The *field-name* is the name of the field or group for which the value, range, or count is requested or for which a new value is being provided. The name specified must be two characters in length and must be present in the FDT of the file being read or updated by the Adabas direct call. The name can refer to an elementary, subfield, superfield, multiple-value field, a group, or a periodic group.

A field name that refers to a group results in all the fields within the group being referenced. Use of group names can greatly reduce the time required to process the command. A group name cannot be used if the group contains a multiple-value or variable-length field (no standard length).

For access commands, the same name may be specified more than once. In this case, the field value is returned multiple times.

For update commands, the same name cannot be used more than once (except in the case of multiple-value fields, as explained later in this section).

A subfield, superfield, subdescriptor, or superdescriptor name may be specified for access commands but not for update commands.

The following table illustrates the relationship between allowed single-value field types and the type of command:

Command Type	Field	Subfield	Superfield	DE	SUBDE	SUPERDE	COLDE
Read (L*n*) [1]	yes	yes	yes	yes	yes	yes	yes [3]
Add (N*n*)	yes	no	no	yes	no	no	no
Find (S*n*)	yes	yes	yes	yes	yes	yes	yes
Update (A1/4) [2]	yes	no	no	yes	no	no	no

Notes:

1. Format C may be used with read commands to request the record in its compressed format.
2. A field specified for an update command cannot be repeated, cannot contain a "1-N"-type specification, and cannot specify a group and an element (field or group) contained in the group.
3. A collation descriptor (COLDE) can only be specified in the format buffer of the L9 command and only when the decode option has been specified in the user exit. The value returned is not the index value but the original field value.

Index or Range Notation (*i* [-*j*] Notation)

The following is the field name syntax for selecting multiple-value fields or occurrences of periodic groups:

```
field-name i[-j]
```

where:

i	is the periodic group or multiple-value occurrence
i-j	is the periodic group or multiple-value occurrence range

Periodic group names *must* be followed by a numeric or other appropriate suffix (see the discussions of the *Count Indicator (C)* and the highest occurrence/value indicator *Highest Occurrence/Value Indicator (N)* for more information). Specifying a periodic group name as the field name alone is incorrect syntax.

Multiple-value fields can be specified by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. See *Multiple-Value Fields* for more information.

Example	Selects . . .
ZZ3	the third value of multiple-value field ZZ, the third occurrence of periodic group ZZ, or the first occurrence of the single- or multiple-value field ZZ contained within the third occurrence of a periodic group.
ZZ3-6	the third through sixth values of the multiple-value field ZZ, the third through sixth occurrences of periodic group ZZ, or the first occurrence of the (single or multiple-value) field ZZ contained within the third through sixth occurrences of a periodic group.

Periodic Group References

If a periodic group (or a field within a periodic group) is to be referenced, the specific occurrence to be used must be specified. This is accomplished by appending a one- to three-digit index (value 1-191, leading zeros are permitted) to the name.

Consider the following examples:

Note:
The examples in this section use the Adabas file definitions described in *File Definitions Used in Examples*.

1. An occurrence is identified by the name of the periodic group and the occurrence number.

Example	Selects . . .
GB3	the third occurrence of periodic group GB (fields BA3, BB3, BC3).

2. When selecting fields within one or more periodic groups, the field name and occurrences (values) are used; the group name is implied.

Example	Selects . . .
BB06	the sixth occurrence of field BB.

3. To refer to a range of occurrences of a periodic group (or a field within a periodic group), enter the periodic group or field name, followed by the first and last occurrence numbers separated by a hyphen. The occurrence numbers must occur in ascending sequence; a descending range is not permitted.

Example	Selects . . .
GB2-4	the second through fourth occurrences of periodic group GB, including all fields within these occurrences (BA2, BB2, BC2; BA3, BB3, BC3; and BA4, BB4, BC4).
BA2-4,BC2-4	the second through fourth occurrences of field BA and BC (BA2, BA3, BA4, BC2, BC3, BC4).

Multiple-Value Fields

Multiple-value fields can be specified in the format buffer in two ways: by explicitly identifying a particular value (indexing) or by referencing each value in sequence, letting Adabas assign an index based on the sequence. These two methods apply as well to sub- or superdescriptors derived from multiple-value fields.

Note:
The examples in this section use the Adabas file definitions described in *File Definitions Used in Examples*.

1. Indexing: To refer to a particular value of a multiple-value field, enter a one- to three-digit index (value 1-191, leading zeros permitted) after the field name.

Example	Selects . . .
MF2	the second value of the multiple-value field MF.
MF1, MF10	the first and tenth values of the multiple-value field MF.

   To refer to a range of values for a multiple-value field, specify the first and last values desired, connected by a hyphen, immediately after field name. An ascending range must be specified.

Example	Selects . . .
MF1-3	the first three values of the multiple-value field MF.

2. Sequencing: To refer to multiple-value fields by repeating the field name, first specify the first value, then the second, and so on.

Example	Selects . . .
MF,MF	the first and second values of the multiple-value field MF.
AA,MF,AB,MF,AC,MF	the first three values of the multiple-value field MF and the values for the fields AA, AB and AC, in the order shown.

Both methods of referencing a multiple-value field can be used in the same format buffer. If no occurrence index is specified, an index that is one higher than the last index, proceeding from left to right, is implicitly (or explicitly) assigned. If the last index was specified as "N" or "1-N", referring to the highest

existing occurrence, a new field specification without an explicit index will refer to the *same* field occurrence as the last specification. See *Highest Occurrence/Value Indicator (N)* for more information about the highest occurrence/value indicator N/1-N/NC.

For an update command where *all* format buffer references to a multiple-value field are specified without indexes (i.e., by sequencing), only those occurrences specified will remain in the record; all other occurrences that might exist are deleted. If *any one* reference to a multiple-value field is specified with an index, then only the specified occurrences are changed; all other occurrences remain unchanged.

Multiple-Value Fields within Periodic Groups

If a multiple-value field is within a periodic group, specifying the multiple-value field name implies the periodic group name. The group name, therefore, does not have to be specified; only the group occurrence or occurrence range is required. The general syntax for selecting a multiple-value field value or value range within a periodic group occurrence or occurrence range is:

```
field-name i[-j] (i[- j])
```

where:

i	is the periodic group occurrence
i-j	is the periodic group occurrence range
(*i*)	is the multiple-value field value
(*i-j*)	is the multiple-value field value range

Consider the following examples:

Note:
The examples in this section use the Adabas file definitions described in *File Definitions Used in Examples*.

1. To refer to a multiple-value field contained within a periodic group, enter the occurrence number of the periodic group after the field name, followed by the desired multiple-value field values or range of values enclosed within parentheses.

Example	Selects . . .
CB2(5)	the fifth value of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.
CB2(1-5)	the first five values of the multiple-value field CB in the second occurrence of the periodic group that contains field CB.

2. To refer to either the same multiple-value field or the same ranges of multiple-value fields contained within a range of periodic group occurrences, enter the range of occurrences after the field name, followed by the multiple-value field value or range of values, enclosed within parentheses.

Example	Selects . . .
CB3-5(2)	the second value of multiple-value field CB in each of the third through the fifth occurrences of the periodic group containing field CB.
CB1-2(1-4)	the first four values of the multiple-value field CB in the first occurrence of the periodic group containing field CB, followed by the first four values of CB in the second occurrence of the periodic group.

Count Indicator (C)

To obtain the count of periodic group occurrences, or the count of existing values of a multiple-value field not in a periodic group, specify the periodic group or multiple-value field name followed by "C":

```
field-name C     count for multiple-value field or periodic group "field-name"
```

Example	Selects . . .
GBC	the highest occurrence number (also the occurrence count) in periodic group GB.
MFC	the number of existing values in multiple-value field MF that are not contained in a periodic group.

The count is returned in the record buffer as a one-byte binary number unless an explicit length and/or format is specified (see *Length and Data Format*).

If your Adabas file uses the Adabas 8 extended MU/PE limits, verify that your program reads the occurrence count into a record buffer field with two or more bytes so that it can handle two-byte occurrence count values (for example, FB=’MUC,2,B.’). Adabas returns response code 55 (ADARSP055), subcode 9 if you only provide a one-byte field in the record buffer for the occurrence count of an MU field or PE group in a file with extended MU/PE limits.

To obtain the count of the existing values of a multiple-value field contained within a periodic group, enter the multiple-value field name followed by the group occurrence and "C":

```
field-name i C   count for multiple-value field "field-name" within a group occurrence
```

Example	Selects . . .
CB4C	the number of existing values of multiple-value field CB in the fourth occurrence of a periodic group.

For update commands, specifying "C" causes Adabas to skip in the record buffer the number of positions occupied by the count field, thus ignoring the count.

The user cannot directly update multiple-value or periodic group count fields. These count fields are updated by Adabas when multiple-value field values and periodic group occurrences are added or deleted.

Daylight Savings Indicator (D)

If a user session time zone uses daylight savings time, the user's local time requires a daylight savings indicator to ensure that the accurate time is reported during the hour when time is turned back from daylight savings time to standard time. Specifying the daylight savings indicator (D) on a date-time field can be used to provide you with information about whether the field's date-time value is in standard or daylight savings time.

The syntax of the daylight savings indicator is simply the letter "D" specified after the field name and before any optional MU index, PE index, or MU-PE index combinations:

```
fnD[mu-pe-index]
```

The daylight savings indicator can only be specified for date-time fields defined with the TZ option. When a daylight savings indicator is specified for a field in a format buffer, it must have a format of 2,F. The offset from standard time (in seconds) is specified in the record buffer associated with the format buffer in a halfword:

```
H'nnnn'
```

The value *nnnn* represents the number of seconds that the stored time should be offset from standard time to calculate daylight savings time. A value of zero indicates that standard time should be used; any value other than zero indicates that daylight savings time should be adjusted for and specifies the offset for that adjustment.

It is possible that the daylight savings time offset is negative if the previous standard time belongs to a different time zone and the regrouping coincided with the change to daylight savings time. In addition, any security-by-value criteria on a date-time field defined with the TZ option will be evaluated as given in UTC time.

Valid MU and PE Index Specifications in the Format and Search Buffers

Valid MU and PE index specifications in daylight savings indicator syntax can be any of the following:

- An MU index for an MU field
- A PE index for a PE group
- A range of MU indices
- A range of PE indices
- A PE index or range of PE indices and an MU index or range of MU indices for an MU field within a PE group.

In other words, all of the following are allowed:

Specification	Description
i	MU index for an MU field or a PE index for a PE group.
i-j	Range of MU indices or PE indices.
N	The highest MU index for an MU field or PE index for a PE group.
1-N	Range of all MU or PE indices.
***i*(*m*)**	The MU index (*m*) for an MU field in the PE group identified by the PE index (*i*).
***i*(*m*-*n*)**	A range of MU indices (*m*-*n*) for an MU field in the PE group identified by the PE index (*i*).
***i*(N)**	The highest MU index (N) for an MU field in the PE group identified by the PE index (*i*.
***i*(1-N)**	All MU indices (1-N) for an MU field in the PE group identified by the PE index (*i*.This specification is not allowed for update commands.
N(*m*)	The MU index (*m*) for an MU field in the PE group identified by the highest PE index (N).
N(*m*-*n*)	The range of MU indices (*m*-*n*) for an MU field in the PE group identified by the highest PE index (N). This specification is not allowed for update commands.
N(N)	The highest MU index (N) for an MU field in the PE group identified by the highest PE index (N).
N(1-N)	All MU indices (1-N) for an MU field in the PE group identified by the highest PE index (N). This specification is not allowed for update commands.
i-j(m)	The MU index (*m*) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).
i-j(m-n)	The range of MU indices (*m*-*n*) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).
i-j(N)	The higheset MU index (N) for an MU field in the PE group identified by the range of PE indices (*i*-*j*).
i-j(1-N)	All MU indices (1-N) for an MU field in the PE group identified by the range of PE indices (*i*-*j*). This specification is not allowed for update commands.

For example, assuming field AA is a date-time field (or a PE group containing a date-time field), any of the following daylight savings indicator specifications might be valid:

- AAD -- Daylight savings time adjustments should be applied to field AA.
- AAD1 --Daylight savings time adjustments should be applied to occurrence 1 of MU field AA or PE field AA.
- AAD2(5) -- Daylight savings time adjustments should be applied to occurrence 5 of the multiple-value field AA in the second occurrence of the periodic group including field AA.

Daylight Savings Indicator Rules

The following rules apply to the use of a daylight savings indicator:

1. Date-time edit masks and the daylight savings indicator are not allowed in the field selection criteria of a format buffer.
2. Fields with the daylight savings indicator specified in the format buffer must have formats of 2,F.
3. The daylight savings indicator can be specified only for fields defined with the TZ option.
4. Each date-time field with a daylight savings indicator in a format buffer must also be specified alone (without the daylight savings indicator) in the format buffer. For example:

   ```
   "AA,14,U,AAD,2,F."
   ```

5. If the field for which a daylight savings indicator is specified is a multiple value field, the older MU syntax is not supported. For example, instead of specifying the MU syntax "AAD,AAD,AAD", you must specify "AAD1-3".
6. The daylight savings indicator cannot be specified more than once for the same field, MU occurrence or MU start occurrence.
7. The following rules apply in update operations::
   - The daylight savings indicator is evaluated for times where the daylight savings time is set back to disambiguate between date-time values and when advancing the clock for daylight savings time to detect invalid nonexistent local times.
   - If the daylight savings indicator is omitted, Adabas assumes that standard time is used in cases whether the date-time value might be ambiguous.
8. In search buffers, the following rules apply:
   - A daylight savings indicator cannot be specified without a corresponding date-time field. The date-time field must be defined with one of the following date-time edit masks: DATETIME, TIMESTAMP, or NATTIME).
   - The daylight savings indicator must precede the corresponding date-time field. For example:

     ```
     AAD,AA,14,U,E(DATETIME),GE
     ```

9. When sorting or reading local time values of a field with the TZ option, it may seem that the values do not appear to be in the correct order. Only when combined with the the daylight savings indicator can the local time can be precisely represented: Suppose we have stored the following DATETIME UTC values:

   ```
   2009-11-01 07:40:00
   2009-11-01 08:30:00
   2009-11-01 09:20:00
   ```

   When reading with "TZ=America/Denver", the second value or the response appears to be not in sequence:

   ```
   2009-11-01 01:40:00
   2009-11-01 01:30:00
   2009-11-01 02:20:00
   ```

   Yet the D indicator for the 3 values returns:

   ```
   H’3600’
   H’0’
   H’0’
   ```

   The first time value was when the 1 hour of daylight saving time was active.

Daylight Savings Indicator Example

Suppose the definition of field AA in the FDT is:

```
"1,AA,14,U,TZ,DT=E(DATETIME)"
```

A valid format buffer might be:

```
"AA,14,U,AAD,2,F."
```

The corresponding record buffer might be:

```
"20080814120000",H’3600’
```

In this example the daylight savings offset for field AA is 1 hour (3600 seconds) from standard time.

Length Indicator (L)

The format buffer indicator, *L*, can be used to retrieve or specify the actual length of any LA or LB alphanumeric or wide-character field value. This format buffer element is referred to as the *length indicator*.

Note:
At this time, the length indicator can only be used in format buffer specifications for LA or LB fields. Support for use of the length indicator in other fields as well will be considered in a future release of Adabas.

The length indicator is specified using the field name followed by the character *L* (for example, `FB=’ACL,4,B.’` would return the length of the AC field). If the field is a multiple-value field or is located in a periodic group, the field name and character L are followed by the related occurrence indices (for example, `FB=’ACL2,4,B.’` would return the length of the second value of multiple-value field AC). The compressed field length is returned in four-byte binary format. A different length and format

cannot be specified.

- Using the Length Indicator with MU/PE Fields
- Using the Length Indicator in Read Commands
- Using the Length Indicator in Update Commands

Using the Length Indicator with MU/PE Fields

When used with MU or PE fields, the length indicator must specify occurrence indices, including the range of occurrence index. However, it cannot specify the 1-N occurrence index. Consider the following examples.

1. In the following example, the length of the fifth value of the second occurrence of periodic group field AC would be returned:

   ```
   FB='ACL2(5).'
   ```

2. In the following example, the lengths of the first ten values of multiple value field AC would be returned:

   ```
   FB='ACL1-10.'
   ```

3. The following example is illegal as the 1-N notation is notation is not allowed with the length indicator in MU/PE fields:

   ```
   FB='ACL1-N.'
   ```

4. The following example is also illegal as the length indicator does not support MU fields with out an occurrence index:

   ```
   FB='MCL.'
   ```

In addition, if you elect to combine the length indicator and an asterisk length notation value request in the same format buffer for an MU or PE field, the value requests must use corresponding ranges as the length requests. It does not matter whether the length requests and value requests are specified in the same or different format buffer segments. Consider the following examples, where XX is an LA or LB field with the MU option:

1. The following valid examples request the length of the first two values of the XX field as well as their actual values.

   ```
   FB='XXL1-2,XX1-2,*.'
   ```

   ```
   FB='XXL1,XXL2,XX1,*,XX2,*.'
   ```

2. The following *invalid* examples are attempts to request the length of the first two values of the XX field as well as their actual values. However, these examples are invalid because the ranges specified for the MU field in the length and value requests are not specified in a corresponding manner.

   ```
   FB='XXL1,XXL2,XX1-2,*.'
   ```

   ```
   FB='XXL1-2,XX1,*,XX2,*.'
   ```

3. The following two format buffers request the length of the third and fourth values of the XX field, as well as their actual values.

   ```
   FB='XXL3,XXL4.'

   FB='XX3,*,XX4,*.'
   ```

4. The following invalid format buffers attempt to request the length of the third and fourth values of the XX field, as well as their actual values, but fail because the ranges specified for the length and value requests are not specified in a corresponding manner.

   ```
   FB='XXL3,XXL4.'

   FB='XX3-4,*.'
   ```

Using the Length Indicator in Read Commands

When the length indicator is specified for a field in the format buffer of a read command, the number of bytes required for the field value in the record buffer (without padding and with no further length indication) is returned at the corresponding field position in the record buffer. The amount of space required in the record buffer is based on the field format and the UES-related definitions for the database, file, and user.

You cannot, in the same format buffer, specify the length indicator to retrieve the length of an LA or LB field in combination with a request for the actual field value in a different format (converted) from the field's base format. For example, if character LB field L1 is stored in format A, `FB='L1L,4,B,L1,*,W.'` is an illegal format buffer specification because it requests the length of the the L1 field in addition to the value of the L1 field converted to Unicode (format W). The reason for this restriction is that the length element gives the length of the field in its native format, but the length of the value returned in the requested format (Unicode) would be different due to the conversion.

If character LB field L1 (format A) contains a 40,000-byte EBCDIC value, consider the following examples:

1. Suppose the format buffer specification for L1 is:

   ```
   FB='L1L,4,B.'
   ```

   The record buffer will contain the four-byte binary length of the value of field L1:

   ```
   X'00009C40'
   ```

2. Suppose the format buffer specification for L1 is:

   ```
   FB='L1L,4,B,L1,*,A.'
   ```

   The record buffer will contain the four-byte binary length of the value of field L1 at the beginning of the record buffer area , followed by 40,000 characters of the actual L1 data.

Empty Values, Length Indicators, and the NB (No Blank Compression) Option

If the length indicator of a field is specified in the format buffer (for example, `FB='L1L,4,B.'`), and if the field is *not* subject to blank compression (the NB option is specified for the field in the FDT), the length returned is the number of bytes specified when the value was stored (which can be zero). However, if the field is subject to blank compression, the length returned is the number of significant left-most

bytes, beyond which the value is padded with blanks. For an all blank value, the returned length is the length of one blank (one byte for alphanumeric fields, two bytes for wide-character fields).

The following examples depict this. For the first three examples, suppose the FDT is defined as follows:

```
1 | Z1 | 022 | A | NU
1 | Z2 | 000 | A | LB NU
1 | Z3 | 022 | A | NU
```

Note:
In Natural, Z2 is a large object (LOB) field with DYNAMIC specified in the DDM. So note that the field length is zero ("000") in the FDT.

Example 1: A Natural application fills all three fields:

```
Z1-FIELD := ’1234567’
Z2-FIELD := ’LOB LOB LOB’
Z3-FIELD := ’89101112’
```

Result: Response Code 0 (ADARSP000).

Example 2: A Natural application fills fields Z1 and Z3:

```
Z1-FIELD := ’1234567’
Z3-FIELD := ’89101112’
```

Result: NAT3052 Error processing a buffer. DB/FNR/Subcode xxx/yyy/2. Invalid length for variable-length field specified in record buffer.

Example 2 failed because an attempt was made to store field Z2 with a length of zero. Adabas assumes that fields cannot be stored with a length of zero unless they are defined with the NB option. If a field is not defined with the NB option (as with Z2 in this case), but an attempt is made to store it with a length of zero, an error results. To resolve this, store the field you want empty as a single blank (as shown in the next example) or define it as an NB field (as shown in Example 4).

Example 3: A Natural application fills fields Z1 and Z3 with values, but fills field Z2 with only the initial value of an Alpha field (blank):

```
Z1-FIELD := ’1234567’
Z2-FIELD := ’ ’
Z3-FIELD := ’89101112’
```

Result: Response Code 0 (ADARSP000).

Now suppose the definition of field Z2 in the FDT is *not* subject to blank compression (the NB option has been added to its FDT definition):

```
1 | Z1 | 022 | A | NU
1 | Z2 | 000 | A | LB NU NB
1 | Z3 | 022 | A | NU
```

In this case, Example 2 above would respond differently, as shown below in Example 4.

Example 4: A Natural application fills fields Z1 and Z3, when Z2 includes the NB option:

```
Z1-FIELD := ’1234567’
Z3-FIELD := ’89101112’
```

Result: Response Code 0 (ADARSP000).

Using the Length Indicator in Update Commands

When a length indicator is specified in the format buffer for an update command, the corresponding value in the record buffer specifies the actual value length of the field in the record buffer. Only one length indicator for the base field can be specified and it must be accompanied by the asterisk (*) length notation in the format buffer.

The length indicator must occur in its format buffer segment prior to any format element that implies a variable length in the record buffer (due to the use of asterisk notation or zero length notation). In other words, the length indicator is located in a constant position, independent of the values of any fields mentioned in the format.

Highest Occurrence/Value Indicator (N)

The indicator "N" selects the last value in a series of values comprising a multiple-value field, or the last occurrence of a periodic group, removing the need to know the number of the last value or occurrence.

Note:
The 1-N notation is not supported for LB fields.

The notation "1-N" selects all values comprising a multiple-value field, or all occurrences of a periodic group. For multiple-value fields in periodic groups, it is not possible to combine the specification 1-N for the group occurrence with any specification for the field occurrences.

The notation "NC" selects the count of the existing values of a multiple-value field in the last occurrence of the periodic group containing the field.

field-name N	Last occurrence of periodic group *field-name* or the highest value of multiple-value field *field-name*.
field-name NC	Count of values for multiple-value field *field-name* in the last occurrence of the periodic group containing *field-name*.
field-name NC	Selects all values of multiple-value field *field-name* or all occurrences of the periodic group *field-name*.
field-name N (N)	The last value of multiple-value field *field-name* in the last occurrence of the periodic group containing *field-name*.
field-name N(1-N)	All values of multiple-value field *field-name* in the last occurrence of the periodic group containing *field-name*.

field-name N(*i* [-	Value or range of values of multiple-value field *field-name* in the last occurrence of the periodic group containing *field-name*.
	
	For multiple value LB fields and LB fields within periodic groups, you must specify a specific occurrence of the field. In the following example, the first value of multiple-value LB field L2 is selected:
	
	`FB='L21.'`
	
	You cannot specify the base field without an occurrence index. For example, the following format buffer specification is *not* valid:
	
	`FB='L2.'`

field-name i [- *j*]	The last value of multiple-value field *field-name* in an occurrence or range of occurrences of the periodic group containing *field-name*. For multiple value LB fields and LB fields within periodic groups, you must specify a specific occurrence of the field. In the following example, the fifth value of LB field L3 in its second PE-group instance is selected: `FB='L32(5).'`
field-name i [-*j*]	All values of multiple-value field *field-name* in an occurrence or range of occurrences of the periodic group containing *field-name*.

Example	Selects . . .
MFN	the highest (last) value of multiple-value field MF. If multiple-value field MF contains four values, the fourth value (the last value entered) is selected.
GBN	the highest occurrence number of a periodic group GB.
MFNC	the count of existing values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MF1-N	all values of the multiple-value field MF.
GB1-N	all occurrences of periodic group GB.
MFN(1-N)	all values for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(N)	the highest value for the multiple-value field MF in the highest occurrence of the periodic group containing MF.
MFN(4)	the fourth value of the multiple-value field MF in the highest occurrence of the periodic group containing MF.
CB3-5(N)	the highest value of the multiple-value field CB in the third through fifth occurrences of the periodic group containing CB.
CB2(1-N)	all values of the multiple-value field CB in the second occurrence of the periodic group containing CB.

SQL Significance Indicator (S)

The "S" significance, or null, indicator and the corresponding null indicator value in the record buffer indicate whether a field’s value is significant, including zero or blank, or not significant (undefined). The S indicator can only be applied to elementary fields that are defined with the NC option, but not for an NU option field:

```
field-name S    SQL significance indicator
```

See the section *Record Buffer* for a description of the null value indicator, and the ADACMP utility description in the *Adabas Utilities* documentation for information about defining SQL null fields.

The related null indicator value in the record buffer, described below, has a two-byte standard length and fixed point format; this length and format cannot be overridden.

For update-type commands, a null value will be stored in the field if the corresponding null indicator value is X‘FFFF’ and no NN option is specified for the field. Otherwise, the null indicator must be X‘0000’. This, combined with the S indicator, shows that the field’s value given elsewhere in the record buffer is to be stored.

The S indicator is not required for update-type commands; if the S indicator is not specified, the field value is updated exactly as if the S indicator had been specified and the corresponding null indicator value had been set to zero (a null in the field is a value). See the examples below.

For read-type commands, the S indicator is required when the NC fields are defined without the NN option. If the S indicator is not present when a read command detects an NC-specified field and the field actually contains a null value, a response code 55 (ADARSP055) is returned.

Example:

This example uses the "field-name S" indicator with the two-byte null indicator in the record buffer to update or add a record, setting field AA equal to the SQL null value and ignoring the value for field AA:

Format Buffer	`AAS , AA , 2 , ... .`	Name of the fields to be read
Record Buffer	`FFFF  123F`	Field values returned by Adabas

For examples showing the use of the SQL significance indicator when a group or range of fields containing an NC field is specified, see the section *The SQL Significance Indicator and Field Series Notation*.

The "field-name S" indicator can be anywhere within the format buffer; that is, it need not precede the corresponding field element. For update-type commands, the format buffer cannot contain more than one element referring to the same field:

AAS.	valid for read/update commands
AA.	valid for read/update commands
AAS,AA,AAS.	valid for read commands only

This means that several format buffer elements referring to the same field cannot be specified for an update-type command:

AA,AA.	causes response code 44 (ADARSP044)
AA,AAS,AA.	causes response code 44 (ADARSP044)

Selecting LOB Values or LOB Value Segments

Complete or partial large object (LB) fields (LOB fields) can be specified using the following syntax for *field-name*:

```
field-name [index-or-range] [(bytenum,LOBlength[,LOBlength2])]
```

The following substitutions can be used in this LOB field syntax:

- A two-character LOB field name should be specified for *field-name*. The LOB field may be an elementary or a multiple-value LOB field and can be located in a periodic group. This is the LOB field whose whole or partial value you intend to read or write.
- If the LOB field is a multiple value field or a field in a periodic group, you must specify the appropriate *index-or-range* notation used for MU or PE fields to identify the specific MU or PE field occurrence you want to read or write. If you only want a partial LOB value (if you specify the `(bytenum,LOBlength[,LOBlength2])` construct), a single occurrence of an MU or PE LOB field must be specified; in this case the range notation is not allowed.

- The (*bytenum*,*LOBlength*[,*LOBlength2*]) construct is optional and is known as *LOB segment notation*. It can be used to select only part (a segment) of a LOB value to be read or written. If this construct is omitted, the entire LOB field value is read or written. To use this construct, the field name must defined with the LB (LOB field) option.

  You cannot specify the same LOB field (or the same instance of an MU or PE LOB field) more than once in the same format buffer (with our without LOB segment notation) in a single store or update command (N1 or A1 command).

- Valid values for *bytenum* in LOB segment notation are unsigned decimal numbers or an asterisk (*). If a number is specified, it must lie in the range from 1 through 2,147,483,647. If a number is specified, it denotes the starting position of the byte within the LOB value where the LOB segment read or write should begin. The first (leftmost) byte of an entire LOB value is byte number one (1).

  If an asterisk (*) is specified, it denotes the current position in the LOB value. The current position of a LOB value is tracked in the ISN Lower Limit (ACBISL or ACBXISL) field of the Adabas control block, which contains the number of bytes preceding the segment you now want to read or update. The asterisk notation is valid in A1 and L1/L4 commands that also have their Command Option 2 (ACBCOP2 or ACBXCOP2) fields set to "L". It is useful for reading or writing a LOB value using multiple calls with a single, constant format, as it avoids the need to adjust the *bytenum* value in the format for each call in the sequence. At the end of each of these calls, Adabas returns in the ISN Lower Limit field the byte number of the last byte of the LOB segment just processed. Only one format element can be specified that uses the asterisk positioning notation in all sections of the format buffer.

- Valid values for *LOBlength* in LOB segment notation are unsigned decimal numbers ranging from 1 to 2,147,483,647. It specifies the length, in bytes, of the LOB value segment that should be read or written. Space for a LOB value segment of this size must be provided at the corresponding position in the record buffer segment. If a numeric value is specified for *bytenum* in LOB segment notation, the sum of the byte number referenced by *bytenum* and the value of *LOBlength* must be less than or equal to 2,147,483,647.

- Valid values for *LOBlength2* in LOB segment notation are unsigned decimal numbers ranging from 1 to 2,147,483,647. *LOBlength2* is optional, and can only be specified in write operations. When it is specified, its value must equal the value of *LOBlength*. *LOBlength2* indicates that the LOB value segment in the record buffer should replace a segment of the same size in an existing LOB value, without deleting any existing remainder of the value.

- Length and format overrides are *not* allowed in format elements for LOB segments (when LOB segment notation is used in the format buffer). For example, the following notations are all *invalid*:

  - `FB = ‘LB(100001,25000),0,A.’`
  - `FB = ‘LB(100001,25000),*,A.’`
  - `FB = ‘LB(100001,25000),25000,A.’`

For complete information about processing partial LOB fields, read *Processing LOB Field Segments*.

Length and Data Format

The length and format parameters are used if a field value is being provided or is to be returned in a length or format different from the standard defined for the field in the FDT. If the length or format parameters are omitted, the field value must be provided or is returned in the standard length and format of the field:

```
[ , length ] [ , data-format ]
```

Possible format and length conversions are suggested by the information in the following table. A format conversion cannot be specified for subfields or subdescriptors; superfields or superdescriptors; or hyperdescriptors.

Fmt	Max Length (in bytes)	Data Type	Compatible Formats
A	253	alphanumeric, left-justified	W
W	253 [1]	wide-character, left-justified	A
A,W	16,381 [1,2]	alphanumeric or wide-character with LA (long alpha) option; left-justified; preceded by optional two-byte binary (inclusive) length	W,A
B	126	binary; right-justified; unsigned	A,F,P,U
F	8	fixed-point; right-justified; signed; two, four, or eight bytes	A,B,P,U
G	8	floating-point; four or eight bytes	none
P	15	packed decimal; signed; positive=A,C,E, or F; negative=B or D	A,B,F,U
U	29	unpacked decimal; signed; positive=A,C,E, or F; negative=B or D	A,B,F,P

Notes:

1. Like an alphanumeric field, a wide-character field may be a standard length in bytes defined in the FDT, or variable length. Any non-variable format override for a wide-character field must be compatible with the user encoding; for example, a user encoding in Unicode requires an even length (max. 252 bytes).
2. Maximum long alpha length if the length (variable field length notation) precedes the field in the record buffer; otherwise, the maximum length is 16381 bytes.
3. For LA and LB fields only, you can specify an asterisk (*) instead of a length in the format element. For more information, read *Asterisk (*) Length Notation*.

The length specified must be large enough to contain the value in the chosen format, but cannot exceed the maximum length permitted.

If a length of zero is specified, or if *field-name* refers to a variable-length field (no standard length), the value returned by Adabas in the record buffer is preceded by a one-byte binary field containing the length of the value (including the length byte itself). For update commands, you must provide this length byte at the beginning of the record buffer.

If a zero length is specified in the format element, the amount of space available for LB field values in the record buffer is variable and depends on the actual LB field value. In this case, the first four bytes of the LB field value in the record buffer are used to store the actual length of the LB field, including the four-byte length itself (the LB field value length plus four).

The format specified must be compatible with the standard format of the field.

- Conversion between packed/unpacked decimal values and binary is limited to values between 0 and 2,147,483,647.
- Conversion from a numeric format to alphanumeric results in an unpacked value, left justified, without leading zeros and with trailing blanks. For example, the three-byte packed value "10043F" would be converted to "F1F0F0F4F3404040". Value truncation is possible with this type of conversion.

Note:
Length and format overrides are not allowed in format elements for LOB segments (when LOB segment notation is used in the format buffer).

The following additional topics are covered in this section:

- Asterisk (*) Length Notation
- Edit Mask Notation (Read Operations Only)
- Examples

Asterisk (*) Length Notation

For LA and LB fields only, you can specify an asterisk (*) instead of a length in the format element. Asterisks cannot be specified for regular alphanumeric or wide-character fields. The presence of an asterisk indicates that the amount of space available for the LB field value in the record buffer is variable and depends on the actual value of the LB field. However, unlike the zero length specification setting, *no* four-byte length field precedes the LB field value in the record buffer; the record buffer area corresponding to the LB format element only contains the value of the LB field. The actual LB field value length *should be* retrieved for read commands and *must be* specified for update commands using the new format buffer length indicator, *L*. For more information about the length indicator, read *Length Indicator (L)*, elsewhere in this guide.

In the following example, the record buffer for LB field L1 contains only the value of the L1 field, followed by the value of the AA field for which 10 bytes have been allotted.

```
FB='L1,*,AA,10,A.'
```

In the following example, the record buffer for LB multi-value field L2 contains the first ten values of L2.

```
FB='L21-10,*.'
```

The record buffer is not necessarily required to provide sufficient space for the entire field if its format element includes an asterisk length setting. However, in read command processing, the field value can be truncated if *both* of the following conditions are met:

- The record buffer space available is insufficient for the field value.
- A field with asterisk notation is specified at the *end* of the format buffer.

In these conditions, no error is returned. If this were the case in the second example above (`FB='L21-10,*.'`), Adabas would truncate the ten values to be read down to the length of the corresponding record buffer segment. (The truncation occurs from right to left; that is, the last value is truncated first; if the remaining space is still insufficient, the second-to-last value is truncated, and so on.) In extreme cases, if no space is available at all for the field value, the value is truncated down to zero bytes.

In the first example above (`FB='L1,*,AA,10,A.'`), if the record buffer segment is too short, no truncation occurs because this is not allowed for fields specified with a fixed length or length of zero (0). Rather, the nucleus returns response code 53 (ADARSP053), indicating that the record buffer is too small.

Only read commands executed by the Adabas nucleus may truncate values specified with the asterisk notation; no truncation occurs in update commands. In addition, the ADACMP utility does not truncate values specified with the asterisk notation.

Edit Mask Notation (Read Operations Only)

Edit masks are used according to the standard edit mask rules used in the COBOL programming language.

An edit mask may only be specified for numeric fields. All data returned by Adabas to an edited field is converted to unpacked decimal format regardless of the standard format of the field. A maximum of 15 digits (not counting edit characters) can be returned to an edited field.

For a field with an edit format specified, the length parameter must be large enough to contain the field value plus all required edit characters.

Format	Generates the edit mask . . .
E1	zzzzzzzzzzzzzzz
E2	zzzzzzzzzzzzz9-
E3	zzzzzzzzz99.99.99
E4	zzzzzzzzz99/99/99
E5	z.zzz.zzz.zzz.zzz,zz
E6	z,zzz,zzz,zzz,zzz.zz
E7	z,zzz,zzz,zzz,zz9.99-
E8	z.zzz.zzz.zzz.zz9,99-
E9	*,***,***,***,**9.99-
E10	*.***.***.***.**9,99-
E11	user-designated mask
E12	user-designated mask
E13	user-designated mask
E14	user-designated mask
E15	user-designated mask

Note:
Although edit formats E3 and E4 provide space for the century digits (see the following examples), they do not *enforce* date formats that are compatible with year 2000 requirements.

For information on date-time edit masks, supplied with Adabas, read *Date-Time Edit Mask Reference*. Date-time edit masks can be used for record updates, in addition to reads. For information on how these date-time edit masks can be used and processed in format and search buffers, read *Date-Time Edit Mask Processing in Format and Search Buffers*.

Examples

Format Buffer	Field Value	Edited Value
XC,15,E1.	009877	bbbbbbbbbbb9877
XC,8,E4.	301177	30/11/77
XB,5,E7.	-366	3.66-
XB,7,E9.	542	**5.42
Y2,10,E4.	20000229	2000/02/29

Field Series Notation

The notation "field-name - field-name" may be used to refer to a series of consecutive fields (as ordered in an FDT). The user specifies the beginning and ending field names connected by a dash:

```
field-name - field-name  series notation
```

No multiple-value field or periodic group may be contained within the series.

A name that refers to a group may not be specified as the beginning or ending name, but a group may be embedded within the series.

Standard format and length is in effect for all the fields within the series. No length or format override is permitted.

Example	Selects . . .
AA-AC	the fields AA, AB, and AC.
AA-GC	nothing. The series may not contain a multiple-value field or a periodic group.
GA-AC	nothing. The series may not begin/end with a group.
AA,5,U,-AD	nothing. A length and/or format override is not permitted in a series notation.

The SQL Significance Indicator and Field Series Notation

When a group or range of fields contains a field specified with the NC option, the corresponding S operator is optional for read (L*x*) commands. For update (A1) commands, the S operator must not be specified. Adabas assumes that the null indicator corresponding to the NC field in the format buffer is located just in front of the field's value in the record buffer.

For example, given the following field definitions in the FDT:

```
01,GR
  02,AA,8,A
  02,BB,8,A,NC
  02,CC,8,A
```

if the format buffer of an update-type command specifies GR. or AA-CC. , the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
```

That is, the null indicator must be included in the record buffer sequence, although the S indicator was not (and must not be) specified in the format buffer.

If the format buffer of a read (L*x*) command specifies GR,BBS. or AA-CC,BBS., the record buffer has the following structure:

```
AA-value null-indicator-BB BB-value CC-value
null-indicator-BB
```

In other words, the first appearance of the null indicator is implied in the record buffer while the second appearance was explicitly called for by the format buffer.

Space Notation (nX)

The nX syntax is used differently for read and update commands:

```
nX
```

For read commands, nX indicates that "n" spaces are to be inserted in the record buffer by Adabas immediately before the next field value:

Format Buffer	`AA , 5X , BB`	Name of the fields to be read
Record Buffer	`value-AA 5-blanks value-BB`	Field values returned by Adabas

For update commands, nX causes "n" positions in the record buffer to be ignored by Adabas:

Format Buffer	`XX , 5X , YY`	Name of the fields to be updated
Record Buffer	`value-XX ignore-5-bytes value-`	Field values provided by user

Text Insertion Notation

The text syntax is used differently for read and update commands:

```
'text'
```

For read commands, the character string specified in the format buffer is to be inserted in the record buffer immediately before the next field value. The character string provided can be 1-255 bytes long, and may contain any alphanumeric character except a quotation mark.

For example:

Format Buffer	`AA,'here is some text', BB`	Name of the fields to be read
Record Buffer	*value-AA* `here is some text` *value*	Field values returned by Adabas

For update commands, the number of positions enclosed within the apostrophes in the format buffer will be ignored in the corresponding positions of the record buffer.

Specifying Field Lengths of LA (Long Alpha) Fields in Format Buffers

The LA option is normally used with variable-length data. The length of an alphanumeric field with the LA option can also be specified in the format buffer. However, the length is then limited to 16381 bytes:

Format Buffer	`AA,5, ...`	Name of the fields to be read
Record Buffer	`ABC__`	Field values returned by Adabas

A field with LA option can also have the NU (null suppression), NC/NN SQL null significance, or the MU (multiple-value field) option, and can be a member of a PE (periodic) group.

A field with the LA option cannot be a descriptor and cannot be the parent of subfields, superfields, subdescriptors, superdescriptors, hyperdescriptors, or phonetic descriptors.

A field is compressed the same way, with or without the LA option; that is, by removing trailing blanks. This must be kept in mind if you store binary data in an LA field.

Format Buffer Performance Considerations

Performance improvements may be achieved by using the following guidelines during format buffer construction:

- Use group names wherever possible rather than referring to elementary fields individually. Use of group names reduces the time required by Adabas to interpret the format buffer.
- Use of the field series notation does not result in performance improvements. A field series notation is converted by Adabas into a series of elementary fields.
- Use length and format overrides only when necessary. Using overrides requires additional processing time when interpreting the format buffer and when processing the field.
- If the same fields of a record are to be read and then updated, the same format buffer should be used for the read and update commands. For more information, refer to the descriptions of command and format IDs.
- You should request periodic (PE) group and multiple-value (MU) field occurrences, based on the requirements of the application. In other words, do not arbitrarily request all occurrences; doing so requires extra time to translate the format buffer, and may mean decompressing numerous empty occurrences. Generally, the AA1-N field argument is the most efficient for selecting periodic group occurrences.