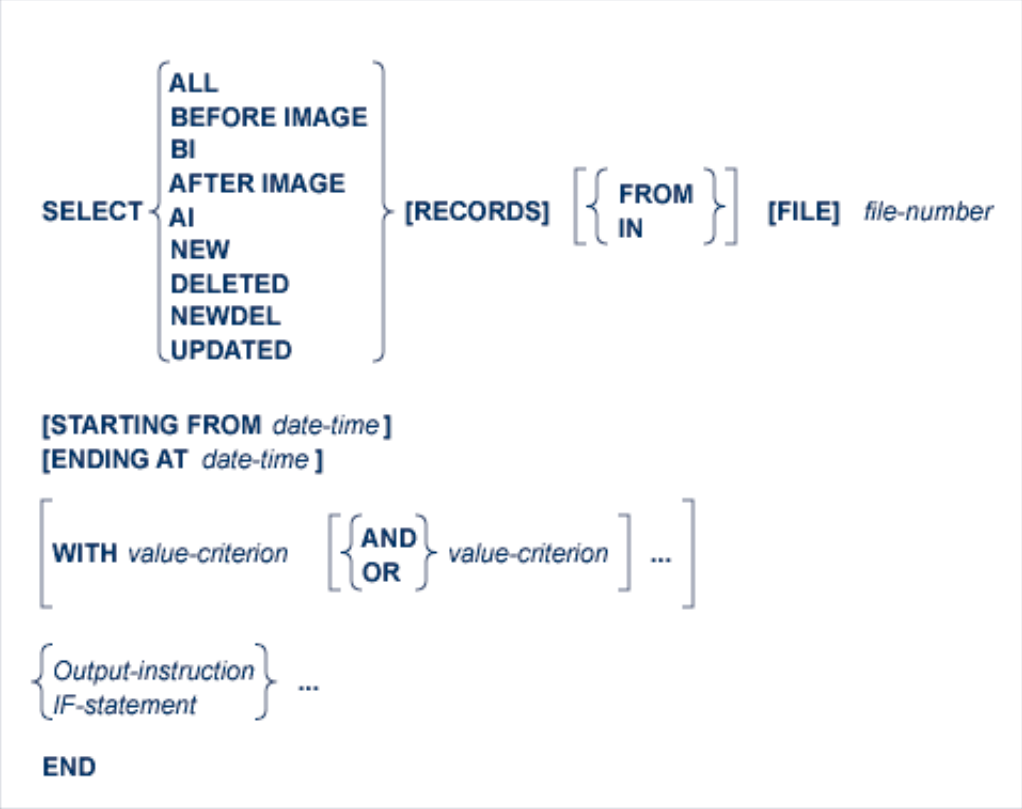
ADASEL Syntax

Unlike other Adabas utilities, ADASEL does not require the utility name at the beginning of each parameter line. A selection request includes the following parts:

- the keyword *SELECT*, followed by the selection option and the file number;
- optional clauses and statements that specify additional selection criteria;
- one or more output instructions;
- the keyword *END*.

An overview of the ADASEL syntax is shown below.



You can code multiple selection requests. Each request begins with SELECT and ends with END.

Example:

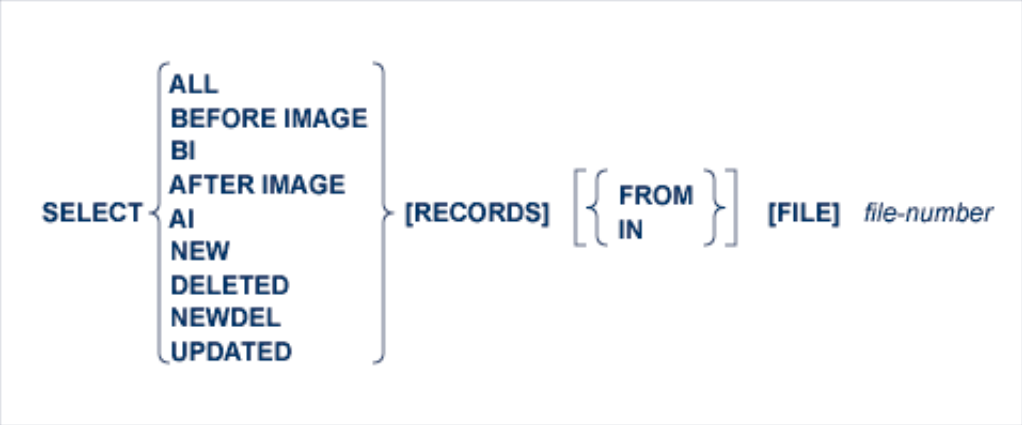
```
SELECT ALL RECORDS FROM FILE 1
   DISPLAY AA BB CC
END

SELECT BEFORE IMAGE FILE 2
   OUTPUT TO EXPA1
END
```

This chapter covers the following topics:

- SELECT Statement
- Additional Selection Criteria
- date-time
- WITH Clause
- IF Statement
- value-criterion
- output-instruction

SELECT Statement



After the keyword SELECT, specify one of the following selection options:

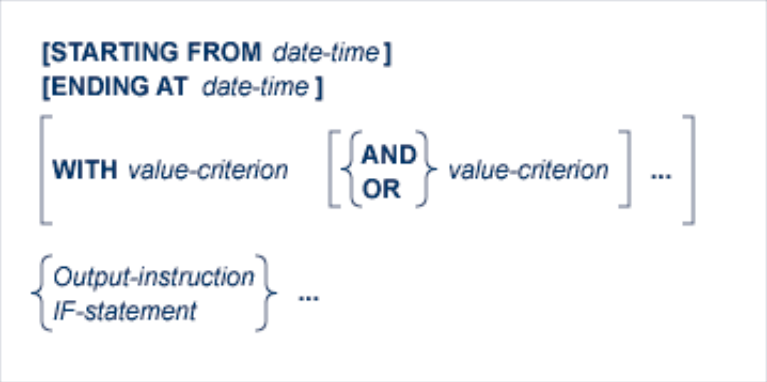
Option	Records Selected
ALL	Before-images derived from A1 (update) and E1 (delete) commands; after-images derived from A1 and N1 (add) commands.
BEFORE IMAGE | BI	Before-images derived from A1 and E1 commands.
AFTER IMAGE | AI	After-images derived from A1 and N1 commands.
NEW	After-images derived from N1 commands.
DELETED	Before-images derived from E1 commands.
NEWDEL	After-images derived from N1 commands and before-images derived from E1 commands.
UPDATED	Before-images and after-images derived from A1 commands.

file-number

Specify the Adabas file for which protection log data is to be selected. Valid file numbers are 0-5000 or 0 through one less than the ASSO block size, whichever is lower. To select user data written by a C5 command, specify the file number of the checkpoint file.

Additional Selection Criteria

You can use the STARTING FROM, ENDING AT, and WITH clauses and the IF statement to specify additional selection criteria. These optional clauses and statement take precedence over the selection option.



The STARTING FROM and ENDING AT clauses restrict selections to records added, updated, or deleted within a time range. The *date-time* variable is discussed below in section*date-time*.

The WITH clause is used to select records that satisfy the value-criteria specified. Multiple conditions can be specified using the logical operators AND and OR. The WITH clause is discussed in section *WITH Clause*.

The IF statement is used to select records and execute output instructions on a conditional basis. The IF statement is discussed in section *IF-statement*.

The syntax of the *value-criterion* variable used in both the WITH clause and the IF statement is described in section value-criterion.

Output instructions are described in section output-instruction.

date-time

The following are valid formats for the *date-time* variable:

Format	Description
yyyymmdd hhmmss	date/time
J(yyyyddd hhmmss)	Julian date/time
X ’ *xxxxxxxx* ’	store-clock (STCK) representation

Note:
The lowest valid value for *yyyy* is "1980".

Examples:

Select all records from file 1 that were added, deleted, or updated on or before midnight of May 12, 1996 (Julian date 132):

```
SELECT ALL RECORDS FROM FILE 1
   ENDING AT J(1996132/240000)
   DISPLAY AA BB CC
END
```

Select all records from file 112 that were added, deleted, or updated on or between January 1 and December 31, 1996:

```
SELECT ALL 112
   STARTING FROM 19960101/000000
   ENDING AT 19961231/240000
   OUTPUT TO EXPA1
END
```

WITH Clause

The WITH clause is used to select records that satisfy the value-criteria specified. Multiple conditions can be specified using the logical operators AND and OR.

- If value-criteria are connected by the AND operator, *each* condition must be satisfied in order for the record to be selected.
- If value-criteria are connected by the OR operator, the record is selected if *any* of the conditions is satisfied.

The syntax of the *value-criterion* variable is described in section *value-criterion*.

Example:

The protection log contains before- and after-images for two updated records. The contents of the field BB in the records are shown below:

Before-Image	After-Image
BB = SMITH	BB = ZINN
BB = SMITH	BB = JONES

The SELECT statement includes a WITH clause that further qualifies the selection:

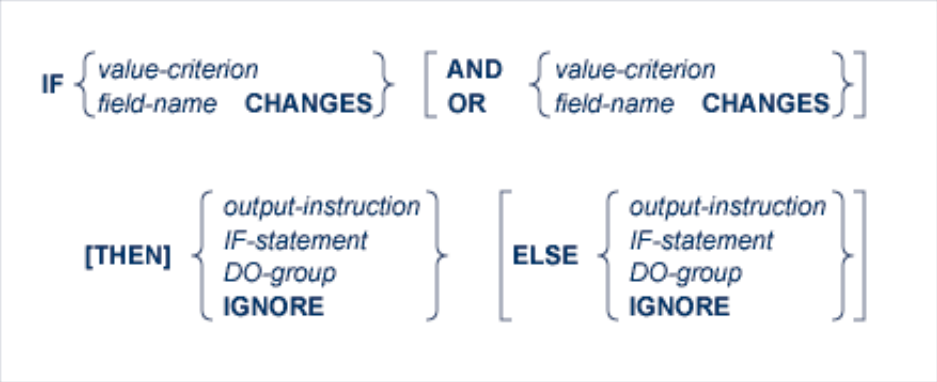
```
SELECT ALL RECORDS FROM FILE 1
   WITH BB ='SMITH'
   DISPLAY AA BB CC
END
```

In this example, despite the fact that the ALL option is used, only the two before-images are selected (because the BB field contains "SMITH" in the before-images). ADASEL ignores all records (in this case, the two after-images) in which the BB field has a value other than "SMITH". If the AFTER IMAGE option were specified, no records would be selected.

IF Statement

The IF statement is used to select records and execute output instructions on a conditional basis.

By default, ADASEL permits up to 20 nested IF statements. For information about changing the default, see the section Overriding ADASEL Defaults with Global Parameters Overriding ADASEL Defaults with Global Parameters.



The syntax of the *value-criterion* variable is described in section value-criterion . Output instructions are described in section output-instruction.

A "*field-name* CHANGES" criterion selects records in which the value of a specified field changed during an update. ADASEL detects the change between the before-image and the after-image. Thus, this criterion is valid only for the A1 (UPDATE) command, which writes both a before-image and an after-image to the protection log. The *field-name* must be the two-character Adabas name of an elementary field in the FDT. It *cannot* refer to a group, periodic group (PE), superdescriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value field (MU) or a member field of a periodic group (PE); see the section value-criterion, particularly in the subsection Indexes for MUs and PE Member Fields .

Note:
Only the after-image is reported for "IF *field-name* CHANGES" criterion. If you want to report both the before-image and the after-images of a changed field using ADASEL, specify the LOGINFO or EXTENDED options on the OUTPUT instruction for the run. For more information, read *OUTPUT Instruction*.

The syntax for DO group is as follows:

A DO group is a sequence of output instructions (NEWPAGE, SKIP, DISPLAY, and OUTPUT). The group must begin with the keyword DO and end with the keyword DOEND. A DO group cannot contain nested IF statements and cannot be nested within another DO group.

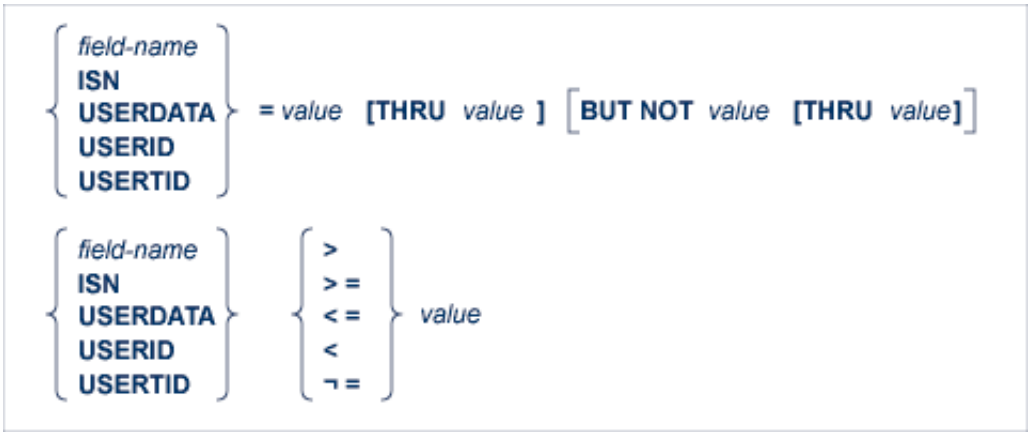
IGNORE instructs ADASEL not to display or output an item.

Example:

```
SELECT ALL FROM FILE 77
   IF AA =’SMITH’ THEN
      IF BB CHANGES THEN DO
         DISPLAY ’Field BB changed:’ BB AA CC
         SKIP 1 LINE
      DOEND
      ELSE DISPLAY AA BB CC
   ELSE IGNORE
END
```

value-criterion

The value-criterion is used in a WITH clause or an IF statement to select records on the basis of a value or values. It has the following syntax:



The BUT NOT clause excludes a value or subrange of values from the range specified in the equality (=).

Object of the Comparison

ADASEL can compare a value or range of values to the

- contents of the specified field. The *field-name* must be the two-character Adabas name of an elementary field in the FDT. It *cannot* refer to a group, periodic group (PE), superdescriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value

field (MU) or a member field of a periodic group (PE); see Indexes for MUs and PE Member Fields .

- ISN; that is, the Adabas internal sequence number of the record.
- USERDATA; that is, the user data written by a C5 command.
- USERID; that is, the user ID (ETID) of the user who added, deleted, or updated the record.
- USERTID; that is, the terminal ID of the user who added, deleted, or updated the record.

Logical Operator

You can express logical operators for equalities and inequalities in words, abbreviations, or symbols as shown in the following table:

Comparison	Words	Abbreviation	Symbol
Equals	EQUAL	EQ	=
Greater than	GREATER THAN	GT	>
Greater than or equal to	GREATER EQUAL	GE	> =
Less than or equal to	LESS EQUAL	LE	< =
Less than	LESS THAN	LT	<
Not equal to	NOTEQUAL	NE	¬=

Note:
The hexadecimal representation of the ¬= symbol is X’5F7E’.

Format of the Value

The format of the criterion value depends on the *default format* of the item that is the object of the comparison.

The default format of an Adabas field (*field-name*) is the format specified in the FDT. The following table shows the maximum length (in bytes) and valid formats for expressing the criterion value:

Criterion Value		Max. Bytes	Max. Digits
Field Format in FDT	**Valid Formats**		
Alphanumeric	Alphanumeric	253	
	Hexadecimal	253	506
Decimal (Packed or Unpacked)	Decimal digits (0-9)	29 *	
Binary	Decimal	4 *	10
	Hexadecimal	126	252
Floating-Point	Hexadecimal	8	16
Fixed-Point	Hexadecimal	4	8
Wide-character	Hexadecimal	253	506

** Excluding minus sign*

The default formats and maximum lengths (in bytes) for other items are as follows:

Item	Default Format	Criterion Value	
		Valid Formats	**Max. Length**
ISN	Binary	Decimal, hexadecimal	4
USERDATA	Alphanumeric	Alphanumeric, hexadecimal	30
USERID	Binary, alphanumeric	Decimal, hexadecimal	8
USERTID	Binary, alphanumeric	Decimal, hexadecimal	8

Value Format Example 1:

If the default format is alphanumeric, the value can be expressed in alphanumeric or hexadecimal format.

```
BA EQ 'SMITH' or BA EQ X'E2D4C9E3C8'
```

Value Format Example 2:

If the default format is packed or unpacked decimal, the value is expressed in decimal digits (0-9). A leading minus sign indicates a negative value. Up to 29 digits (excluding the minus sign) are permitted. Other special characters ($, decimal points, commas, etc.) are not permitted.

```
NU = 123456789
NU = -987654321
```

Value Format Example 3:

If the default format is binary, the value can be expressed in hexadecimal or numeric format.

Up to 252 hexadecimal digits (126 bytes) are permitted for a binary Adabas field.

In numeric format, up to 10 decimal digits (4 binary bytes) are permitted. Thus, a binary value expressed in decimal digits can range from -2,147,483,648 through 2,147,483,647.

```
BB = 2147483647  or   BB = X’80000000’
BB = -2147483648 or   BB = X’7FFFFFFF’
```

Alphanumeric Values

Enclose an alphanumeric value in apostrophes:

```
AA =’SMITH’
```

To indicate an apostrophe within an alphanumeric string, use two successive apostrophes with no intervening space or character:

```
JJ =’Smith’’s Market’
```

Hexadecimal Values

Begin a hexadecimal value with an "X" and enclose the value in apostrophes:

```
AA = X’E2D4C9E3C8’
```

A hexadecimal value must have an even number of hexadecimal characters:

```
JJ = X’04D2’
```

Continuation Lines

ADASEL treats columns 1-72 as the input line. To continue an alphanumeric or hexadecimal value on additional lines, place the closing apostrophe only at the end of the entire string. The value is concatenated until the closing apostrophe is found.

In an alphanumeric string, ADASEL includes leading and trailing spaces within apostrophes as part of the string; it ignores them in a hexadecimal string.

Example 1: Alphanumeric String

```
                                                                       7
1.....................................................................2

       AA ='THIS IS AN EXAMPLE OF HOW TO CONTINUE AN ALPHANUMERIC VALU
E. KEY THROUGH COLUMN 72 AND CONTINUE IN COLUMN 1 OF THE NEXT
LINE.'

                                                                       7
1.....................................................................2

       AA ='DO NOT CONTINUE AN ALPHA VALUE THIS WAY. LEADING AND
   TRAILING SPACES IN COLUMNS 1-72 ARE INCLUDED.'
```

ADASEL treats the second value above as follows:

```
'DO NOT CONTINUE AN ALPHA VALUE THIS WAY. LEADING AND         TRAILING BLANKS
IN COLUMNS 1-72 ARE INCLUDED.'
```

Example 2: Hexadecimal String

```
                                                                       7
1.....................................................................2

          XX = X'C1C2C3C4C5C6C7C8C9
                 D1D2D3D4D5D6D7D8D9'
```

ADASEL treats the hexadecimal value above as follows:
`X'C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9'`

Indexes for MUs and PE Member Fields

MU Field or a Member Field of a PE

If the *field-name* refers to an multiple-value field (MU) or to a member field of a periodic group (PE), you must include the index (occurrence number) immediately after the name:

AAi	where "AA" is the field name of an MU and *i* is the index
BBk	where "BB" is a member field of a PE and *k* is the index of the PE

Valid values for *i* and *k* range from "1" through "65,534" if you have Adabas 8 or later installed and if extended MU and PE counts are requested; otherwise the valid values range from "1" through "191".

Note:
The use of more than 191 MU fields or PE groups in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS MUPEX and MUPECOUNT parameters.

Examples:

In file 12, the field JT is an MU. The following statement selects all before-images where the second occurrence of JT is "Programmer":

```
SELECT BI FROM FILE 12
   WITH JT2 = 'Programmer'
   DISPLAY NA
END
```

The field SA is a member of a PE. The following statement selects all records where SA in the third occurrence of the periodic group is greater than or equal to 35000:

```
SELECT ALL FROM 12
   WITH SA3 >= 35000
   DISPLAY NA SA3
END
```

MU Contained Within a PE

If an MU is contained within a PE, *both* indexes (PE and MU) must be specified:

ABk(i)	where "AB" is the name of an MU, *i* is the occurrence of AB, and *k* is the occurrence of the PE to which AB belongs

Example:

In file 211, the multiple-value field ST is a member of a PE. The following statement selects all records in which the third occurrence of ST in the second occurrence of the periodic group is "PAST DUE":

```
SELECT ALL FROM FILE 211
   WITH ST2(3) ='PAST DUE'
   DISPLAY AA BB ST2(3)
END
```

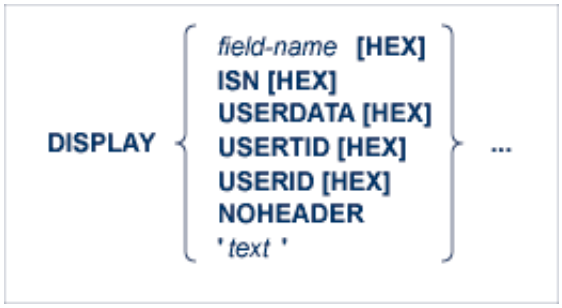
output-instruction

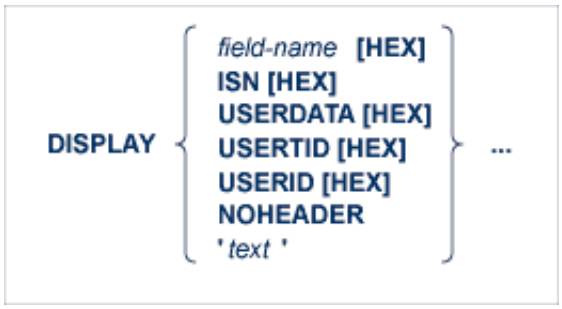
Output instructions include DISPLAY, OUTPUT, SKIP, and NEWPAGE. At least one output instruction is required. Multiple output instructions can be specified, and an output instruction can be included as part of an IF statement. The syntax is shown below:

The DISPLAY instruction is discussed below ; the OUTPUT instruction is discussed in section *Output Instruction*. See also the discussion *SKIP and NEWPAGE*.

DISPLAY Instruction

DISPLAY writes the output report to DDDRUCK/ DRUCK. The syntax specifies one or more output types. When specifying multiple output types, they are separated by at least one space:





where

field-name	displays the contents of the specified field. The *field-name* must be the two-character Adabas field name of an elementary field in the FDT. *field-name cannot* refer to a group, periodic group (PE), superdescriptor, subdescriptor, phonetic descriptor, or hyperdescriptor. However, it can refer to a multiple-value field (MU) or a member field of a PE; indexes for MUs and PE member fields are discussed in section MU or PE Fields.
HEX	displays the hexadecimal value corresponding to the type of output. HEX is especially useful if the output contains unprintable characters. Leave at least one space between the type of output and the following HEX keyword.
ISN	displays the ISN of each selected record.
USERDATA	displays records written to the protection log with a C5 command. The file number of the checkpoint file must be specified in the SELECT statement.
USERID	displays the user ID of the user who added, deleted, or updated the record.
USERTID	displays the TID of the user who added, deleted, or updated the record.
NOHEADER	suppresses the header.
’*text*’	displays the text string.

Examples:

Select records that have been modified. Display the text string "The following records were modified:". Then display the fields AA and CC in hexadecimal format and BB in the format defined in the FDT:

```
SELECT UPDATED RECORDS FROM FILE 117
   DISPLAY ’The following records were modified:’
   DISPLAY AA HEX BB CC HEX
END
```

Display the field AA of each new record, along with the user ID and terminal ID of the user who added the record; suppress the header:

```
SELECT NEW RECORDS FROM FILE 211
   DISPLAY AA USERID USERTID NOHEADER
END
```

Default Formats

A field is displayed according to its default format:

Alphanumeric	is displayed as entered, with unprintable characters converted to blanks.
Binary	is displayed in unsigned decimal digits (0-9) if the value is less than X’80000000’; otherwise, the value is displayed in hexadecimal notation.
Packed/unpacked	is displayed in decimal digits (0-9), with a leading minus sign if the value is negative.

MU or PE Fields

If *field-name* refers to an MU or a member field of a PE, you can display a single occurrence or a range of occurrences by specifying the index as part of the field name:

```
DISPLAY AA5
```

If you have Adabas 8 or later installed and if extended MU and PE counts are turned on for a file, valid index values range from "1" through "65534"; otherwise the valid index values range from "1" through "191". In addition, if you specify "N" as the upper limit of an index range, ADASEL displays all occurrences, beginning with the first occurrence in the range.

Note:
The use of more than 191 MU fields or PE groups in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS MUPEX and MUPECOUNT parameters.

You cannot specify the PE name in a DISPLAY statement. To display the entire periodic group, you must specify the name of each field in the group.

If an MU is contained within a PE, both indexes (PE and MU) must be specified. In the index formats shown below, *i* and *j* are the MU indexes; *k* and *l* are the PE indexes. AB refers to a member field of a PE; MB refers to an MU that is a member field of a PE.

Index	Displays . . .
MU*i*	occurrence *i* of the MU
MU*i*-*j*	occurrences *i* through *j* of the MU
MU*i*-N	all occurrences of the MU, starting with occurrence *i*
AB*k*	field AB in occurrence *k* of the PE to which the field belongs
AB*k*-*l*	field AB in occurrences *k* through *l* of the PE
AB*k*-N	field AB in all occurrences of the PE, starting with occurrence *k*
MBk(*i*)	occurrence *i* of MB in occurrence *k* of the PE to which MB belongs
MB*k* - *l(i)*	occurrence *i* of MB in occurrences *k* through *l* of the PE
MB*k* - *l(i-j)*	occurrences *i* through *j* of MB in occurrences *k* through *l* of the PE
MB*k* - *l*(*i*-N)	all occurrences of MB (starting with occurrence *i*) in occurrences *k* through *l* of the PE
MB*k*-N*(i - j)*	occurrences *i* through *j* of MB in all occurrences of the PE (starting with occurrence *k* of the PE)
MB*k*-N(*i*-N)	all occurrences of MB (starting with occurrence *i*) in all occurrences of the PE (starting with occurrence *k* of the PE)

Example:

File 12 contains the following PE:

Level	Name	Descriptive Name	Format	Length	Options	Occ
1	JT	JOB TITLE	A	16	DE,MU	12
1	PA	INCOME			PE	12
2	SA	SALARY	P	6	DE,MU	7
2	BO	BONUS	P	5		

The following are valid *DISPLAY* statements for file 12:

```
SELECT NEW FROM FILE 12
   DISPLAY JT1
END
```

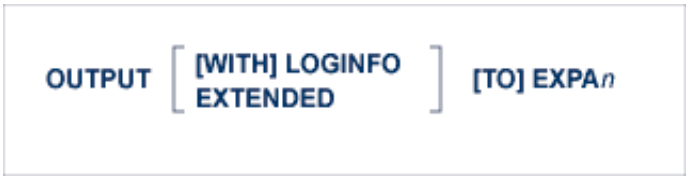
```
SELECT ALL FROM FILE 12
   DISPLAY JT1-5 SA1-5(1-N) BO1-5
END
```

```
SELECT ALL FROM FILE 12
   WITH JT3 =’Programmer’ THRU ’Systems Analyst’
   DISPLAY JT3 SA3(1-N) BO3
END
```

```
SELECT UPDATED FROM FILE 12
   DISPLAY JT2-N SA2-N(1-N)
END
```

OUTPUT Instruction

The OUTPUT instruction is used to write the decompressed records from the protection log to an output data set.



Up to 20 output data sets are permitted. The output data set is specified in the EXPA*n* parameter and the DDEXPA*n*/ EXPA*n* job control statement.

Example:

Write the before-images of all updated or deleted records to data set DDEXPA1/ EXPA1:

```
SELECT BEFORE IMAGE FILE 2
   OUTPUT TO EXPA1
END
```

Output Record Format

The format of the output record depends on whether the LOGINFO or EXTENDED parameter is specified. LOGINFO and EXTENDED are used to display additional information.

Fields common to all output records are shown below. Values in parentheses are field locations when LOGINFO (bytes 32-42) or EXTENDED (bytes 64-74) are specified.

Bytes	Description
0-1	protection log record length (binary)
2-3	set to zeros (X’0000’)
4-5	record image type: C’BI’ before-image C’AI’ after-image C’C5’ user data
6-7	Adabas file number (binary)
8-9 (32-33, 64-65)	decompressed record length (including this length field and the ISN)
10-17 (34-41, 66-73)	ISN (binary) or user data from a C5 command
18 (42, 74)	beginning of the decompressed protection log data

Note:
The first record in each block is preceded by the two-byte block length and two bytes of nulls or blanks.

The fields of the protection log record are provided in the order, length, and format in which they are defined in the file’s FDT. Alphanumeric fields that are longer than the length defined in the FDT are truncated. Numeric fields that are longer than the length defined in the FDT cause ADASEL to end abnormally.

MUs and PEs are preceded by a one-byte binary field containing the number of occurrences.

Variable-length fields have a default length of zero and are preceded by a one-byte field containing the length of the value (including the length field).

If a field defined with the NC suppression option contains a null value, the null value is decompressed by ADASEL to an empty value (blanks or zeros, depending on the field’s format). This type of NC field null processing applies only to ADASEL.

LOGINFO

When LOGINFO is specified, the following additional information is included in each record:

Bytes	Description
8-15	ID of the user who added, deleted, or updated the record
16-19	low-order four bytes of the TID of the user who added, deleted, or updated the record (from the communications ID; TP monitor users only)
20-23	Data Storage RABN where the record was stored (binary)
24-27	data protection block number for the record (binary)
28-31	timestamp of update (binary; high-order four store-clock (STCK) bytes)

EXTENDED

When EXTENDED is specified, the following additional information is included in each record:

Bytes	Description
8-15	ID of the user (ETID) who added, deleted, or updated the record
16-23	low-order eight bytes of the terminal ID of the user who added, deleted, or updated the record (from the communications ID; TP monitor users only)
24-27	Data Storage RABN where the record was stored (binary)
28-31	data protection block number for the record (binary)
32-35	timestamp of update (binary; high-order four store-clock (STCK) bytes)
36	backout indicator: C’B’ record is a result of a backout C’ ’ normal record
37	reserved
38-41	transaction number
42-63	reserved

Output Data Set Designation

The EXPA*n* parameter identifies the output data set. The value of *n* must match the value in the DDEXPA*n*/ EXPA*n* JCL statement. Valid output data set numbers are 1-20 with no leading zeros:

Valid statement	`OUTPUT TO EXPA3`
Invalid statement	`OUTPUT TO EXPA03`

The same rule applies to the DD/EXPAn JCL statement.

Example:

Select all records for file 1. Write decompressed records in which the BA field contains "SMITH" or "SMYTH" to DDEXPA1/ EXPA1. Write all others to DDEXPA2/ EXPA2:

```
SELECT ALL RECORDS FROM FILE 1
IF BA ='SMITH' OR BA ='SMYTH'
   THEN OUTPUT TO EXPA1
ELSE
   OUTPUT TO EXPA2
END
```

NEWPAGE and SKIP Instructions

The NEWPAGE and SKIP instructions control page formatting:

- NEWPAGE forces a page eject before displaying the next line; and
- SKIP prints the specified number of blank lines before displaying the next line of data.

Example:

```
SELECT ALL RECORDS FROM FILE 1
   WITH BA EQUAL 'SMITH' THRU 'SMYTH'
IF BA CHANGES THEN DO
   NEWPAGE
   DISPLAY 'NEW NAME' BA BB BC
   DOEND
ELSE DO
   SKIP 2 LINES
   DISPLAY BA BB BC
   DOEND
END
```