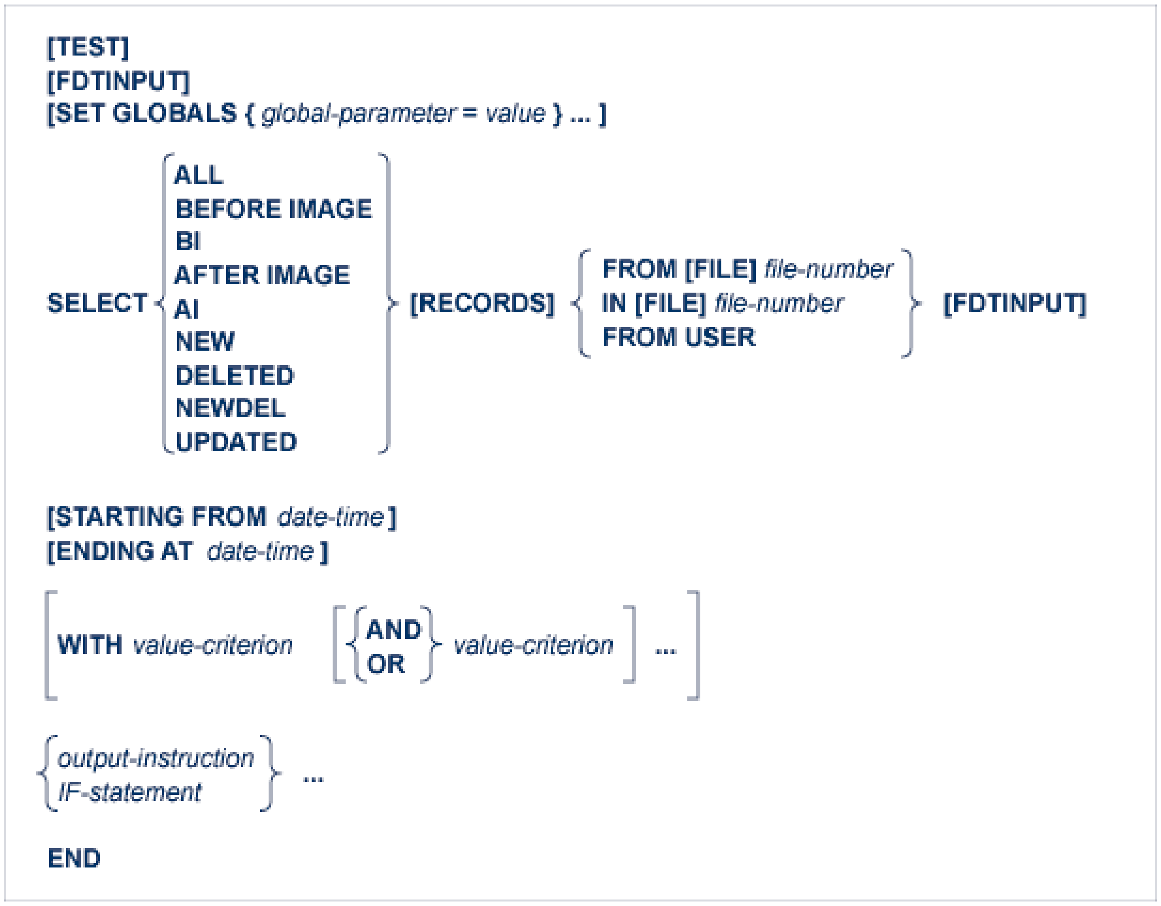
ADASEL Syntax

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

- the keyword SELECT, followed by a selection option and either a file number or the keywords FROM USER
- one or more output instructions or IF statements
- the keyword END.

Optional clauses and other parameters can be included that specify additional selection criteria or processing. The following depicts an overview of the ADASEL syntax.



```
[TEST]
[FDTINPUT]
[SET GLOBALS { global-parameter = value } ... ]

         ALL
         BEFORE IMAGE
         BI
         AFTER IMAGE                      FROM [FILE] file-number
SELECT   AI              [RECORDS]        IN [FILE] file-number       [FDTINPUT]
         NEW                              FROM USER
         DELETED
         NEWDEL
         UPDATED

[STARTING FROM date-time]
[ENDING AT date-time]

[ WITH value-criterion   [ { AND } value-criterion ] ... ]
                             { OR  }

{ output-instruction }  ...
{ IF-statement       }

END
```

Each item in this syntax is now described in the following topics:

- TEST Parameter
- FDTINPUT Parameter
- SET GLOBALS Parameter

- SELECT Parameter

TEST Parameter

The ADASEL utility now includes a syntax-checking-only mode. When the optional TEST parameter is specified, the actual ADASEL utility syntax is checked, but not performed. The TEST parameter should be specified before any other ADASEL parameters, such as any SELECT or SET GLOBALS parameters.

In the following example, the syntax of the SELECT and other parameters will be tested. No actual data will be processed.

```
TEST
SELECT ALL RECORDS FROM FILE 1
   DISPLAY AA BB CC
END
SELECT BEFORE IMAGE FILE 2
   OUTPUT TO EXPA1
END
```

FDTINPUT Parameter

The optional FDTINPUT parameter can be used to indicate that the FDTs used for ADASEL processing should come from an alternate FDT source. This alternate FDT source can be referenced for all selections in an ADASEL run or for individual files or users selected in the ADASEL run. This functionality allows you to handle data situations where the FDT has been modified in some way so it differs from the actual data in the database. In these cases, an older or different FDT might be necessary for ADASEL to accurately process the data in the database.

If FDTINPUT is not specified in an ADASEL run, the FDTs for the files in the database are used by default.

If FDTINPUT is specified in an ADASEL run, a corresponding job statement (DD/SAVE or DD/EBAND) must be specified in the ADASEL run to identify the alternate FDT source to be used, as described in *ADASEL Job Requirements for FDTINPUT*, later in this section.

You can specify the FDTINPUT parameter in an ADASEL run in one of two ways:

1. You can specify it as a global parameter for the ADASEL run, in which case the alternate FDT source identified by the DD/EBAND or DD/SAVE job control statement is used for all files selected and processed in the ADASEL run. In the following example, where FDTINPUT is specified as a global parameter, the FDTs in the alternate FDT source are used for files 20, 35, and 36:

```
FDTINPUT
SELECT ALL FROM FILE 20
   DISPLAY AA BB CC
END
SELECT ALL FROM FILE 35
   OUTPUT TO EXPA1
END
SELECT ALL FROM FILE 36
   DISPLAY ALL
END
```

In the following example, the FDTs in the alternate FDT source are used for all records in the database for user ETID1.

```
FDTINPUT
SELECT ALL FROM USER
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

2. You can specify it separately for individual SELECT statements in an ADASEL run. In the following example, where FDTINPUT is specified for two files, the usual FDT in the database is used for file 20, but the FDTs in the alternate FDT source are used for both files 35 and 36:

```
SELECT ALL FROM FILE 20
   DISPLAY AA BB CC
END
SELECT ALL FROM FILE 35 FDTINPUT
   OUTPUT TO EXPA1
END
SELECT ALL FROM FILE 36 FDTINPUT
   DISPLAY ALL
END
```

Likewise, in the following example, the FDTs in the alternate FDT source are used for all the records in the database for user ETID1.

```
SELECT ALL FROM USER FDTINPUT
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

This section describes the following topics:

ADASEL Job Requirements for FDTINPUT

For ADASEL FDTINPUT processing to be successful, either (but not both) the DD/EBAND or DD/SAVE ADASEL job statements must be specified in the ADASEL run to identify the alternate FDT source that should be used in the run. If neither or both are specified, errors will result. If FDTINPUT is not specified, but either DD/EBAND or DD/SAVE is, a warning message is issued and the job statements are ignored.

- If FDTINPUT is specified for only one file in an ADASEL run, either DD/EBAND or DD/SAVE can be used to identify the alternate FDT source for the run.

- If FDTINPUT is specified for multiple files in an ADASEL run (either on multiple SELECT statements or as a global parameter) or for a SELECT FROM USER selection in an ADASEL run, a DD/SAVE job statement must be used to identify the alternate FDT source for the run. Specifying a DD/EBAND job statement in these instances will result in errors.

- If the alternate FDT source for the ADASEL run is a save tape, the DD/SAVE job statement must be used to identify the alternate FDT source. Specifying a DD/EBAND job statement in this instance will result in errors.

Obtaining an FDT Source for Use with FDTINPUT

Prior to running an ADASEL job with the FDTINPUT parameter, an FDT source must be produced and stored. This can be accomplished in one of two ways:

1. Run the ADAULD utility for a database file that uses the FDT you want to use in your ADASEL run and specify NUMREC=0. This will only unload the FDT. For complete information, read *ADAULD Utility: Unload Files* . FDTs obtained in this manner should be specified in the DDEBAND job statement in the ADASEL job.

2. An old save tape can be used as input to ADASEL FDTINPUT processing. When ADASEL processes the save tape, it reads it sequentially to determine the file numbers of the files on the tape. In addition, if a SELECT FROM USER selection is requested in an ADASEL run, the entire save tape is read in advance to obtain all of the FDTs and their associated file numbers in advance of ADASEL processing.

   To produce a new save tape, run the ADASAV or ADASAV FILES utility function for one or more database files that use the FDTs you want to use in your ADASEL run. This will produce a save tape that can be used as input to the ADASEL run.

In either case, the FDT source provided for FDTINPUT must match the version of the protection log provided in the ADASEL job. In addition, only one alternate FDT source can be specified for FDTINPUT processing in a single ADASEL run.

SET GLOBALS Parameter

ADASEL global parameters override default table and buffer sizes. Overrides are in effect only for the ADASEL run in which the SET GLOBALS statement is specified.

If used, the SET GLOBALS settings must be specified before the first ADASEL SELECT parameter. Comment statements as well as the FDTINPUT and TEST parameters can precede the SET GLOBALS settings. SET GLOBALS settings are specified in the following syntax:

SET GLOBALS { *global-parameter* **=** *value* **} ...**

No spaces are permitted between the parameter name, the equal sign, and the value. However, at least one space must separate parameters. Special characters are not permitted as separators. If multiple lines are used, the SET GLOBALS keyword must be repeated on each line. The first non-blank character string that does not begin with a parameter name terminates the SET GLOBALS statement. Thus, trailing comments are not permitted.

ADASEL provides the following global parameters. Default values are underscored.

Global Parameter	Description
`LPV={`*n* `|` `0}`	Use this parameter to specify the length of the PE-value table used in the evaluation of field values for a PE. Normally, ADASEL uses an estimated number of PE occurrences to compute the table size. If the table size is insufficient, a SEL047 error occurs; you can increase the table size using the global LPV parameter as indicated on the screen.
`LS={`*n* `|` `80}`	Use this parameter to specify the line size parameter is used to alter the number of printed columns. If an output line is longer than the line size, the line is truncated at the nearest blank. The rest of the line is continued on the next output line, beginning in Column 1. The minimum line size is 1; the maximum is 132.
`LST={`*len* `|` `12000 }`	Use this parameter to specify the length of the statement table, which is used to store the translated ADASEL statements. Depending on its complexity, a statement is translated into one or more segments. Each segment is 44 bytes plus a value length. For example: `IF BA EQ 'SMITH'...` requires 49 bytes: 44 bytes plus 5 bytes for "SMITH". The default table size (12 ,000 bytes) handles approximately 200 segments. If the table size is exceeded, a SEL003 error occurs.
`LWP={`*n* `|` `1048576}`	Use this parameter to specify the size of the work pool used internally by the ADASEL utility for spanned record processing. The default value of LWP is 1048576 bytes (or 1MB). If the value specified for LWP is appended with the letter "K", it is multiplied by 1024. Valid values range from 100K - 1048576K (or 1 GB).
`MAXLOGRECLEN={`*n* `|` `1048576`	Use this parameter to specify the size of the uncompressed record buffer allocated by the ADASEL utility for use in spanned record processing. The default value of MAXLOGRECLEN is 1048576 bytes (or 1MB). If the value specified for MAXLOGRECLEN is appended with the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.

Global Parameter	Description

`NCFLD={n | 10}`
 	
`NCUPD={n | 10}`
	Use this parameter to specify the maximum count of "*field-name* CHANGES" statements allowed in the selection query, and the maximum number of parallel updates during the original session. When a statement includes a CHANGES criterion, ADASEL uses a change pool with `NCFLD * NCUPD` entries to track changed field values. If this pool is too small, a SEL060 error occurs. In this case, it is necessary to increase one or both of the parameters and then rerun ADASEL.

`NF={n | 20}`	Use this parameter to specify the maximum number of files that can be processed during a single ADASEL run. NF is used to allocate space for the FDT for each file processed. A SEL014 error occurs if the NF value is exceeded. This value is *not* related to the maximum number of output files (DDEXPA*n*/ EXPA*n*); although more than 20 files can be processed during an ADASEL run, a maximum of 20 output files can be written.
`NIF={n | 20}`	Use this parameter to specify the number of nested IF levels permitted.
`NOUSERABEND`	If specified, ADASEL terminates with condition code 20 instead of a user abend 034 after an error is encountered.
`NU={n | 20}`	Use this parameter to specify the maximum number of user values (or ranges of user values) that can be processed during a single ADASEL run. NU is used to allocate work storage for the user data requested via a SELECT FROM USER specification in ADASEL. The default is 20 user values.

`NV={n | 100}`	Use this parameter to specify the number of field values. NV is used to allocate a table for the evaluation of field values. One entry is required for every field specified in the statements (including duplications). For example, the following statement requires two entries even though the same Adabas field name is used:
	
	`IF BA ='SMITH'`
	`    THEN OUTPUT TO EXPA1`
	`ELSE IF BA ='SMYTH'`
	`    THEN OUTPUT TO EXPA2`

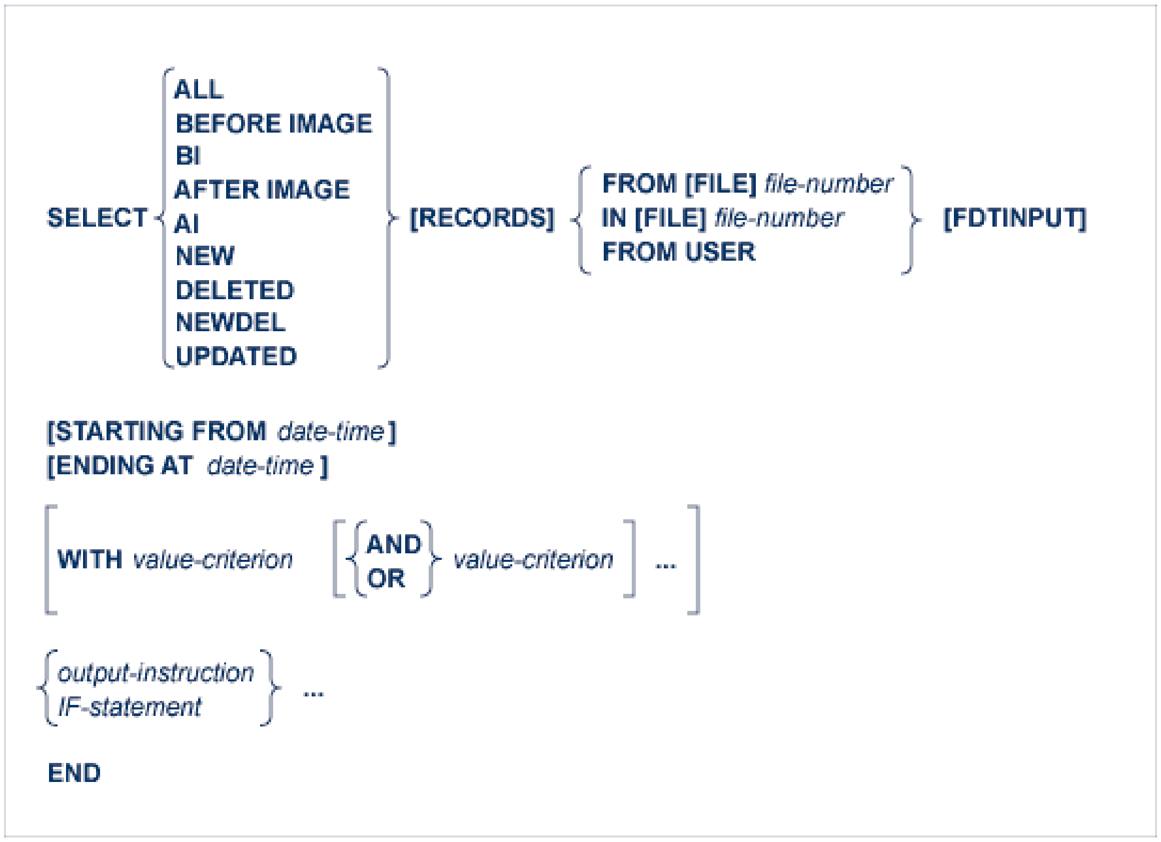
`PS={n | 60}`	Use this parameter to specify the page size parameter is used to alter the number of lines printed before a new page is started. The minimum page size is 2; the maximum is 999.

Example

```
SET GLOBALS LST=15000 NF=15
SET GLOBALS LS=132
```

SELECT Parameter

The syntax of the SELECT parameter is shown below. It begins with a SELECT keyword and ends with the END keyword.



```
         ALL
         BEFORE IMAGE
         BI
         AFTER IMAGE              FROM [FILE] file-number
SELECT   AI              [RECORDS] IN [FILE] file-number    [FDTINPUT]
         NEW                      FROM USER
         DELETED
         NEWDEL
         UPDATED

[STARTING FROM date-time]
[ENDING AT date-time]

[WITH value-criterion  [ AND  value-criterion ] ... ]
                         OR

output-instruction  ...
IF-statement

END
```

The SELECT parameter requires the following minimal specifications:

- the keyword SELECT, followed by a selection option and either a file number or the keywords FROM USER
- one or more output instructions or IF statements
- the keyword END.

Optional clauses and other parameters can be included that specify additional selection criteria or processing. The following depicts an overview of the ADASEL syntax.

This section describes both the required and optional elements of a SELECT parameter, in the order they are shown in the syntax above.

- The SELECT Keyword
- Selection Options (ALL, BEFORE IMAGE, AFTER IMAGE, etc.)
- RECORDS Keyword

- FROM and IN [FILE] file-number Clause
- FROM USER Clause
- FDTINPUT Parameter
- STARTING FROM and ENDING AT date-time Clauses
- WITH Clause
- IF Statement
- value-criterion
- output-instruction
- The END Keyword
- Examples

The SELECT Keyword

The SELECT keyword is a required element of the SELECT parameter and must be specified first.

Selection Options (ALL, BEFORE IMAGE, AFTER IMAGE, etc.)

One of the selection options described in the following table is required in a SELECT parameter and must be specified immediately after the SELECT keyword:

Selection 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.

RECORDS Keyword

The RECORDS keyword is optional in the SELECT parameter. When specified, it should immediately follow the selection option.

FROM and IN [FILE] *file-number* Clause

One of the FROM [FILE], IN [FILE} or the FROM USER clauses is required in a SELECT parameter. Only one is required; if more than one is specified, errors will result. The FROM USER clause is described in *FROM USER*.

Note:
A single ADASEL run cannot include both FROM [FILE] (or IN [FILE]) clauses and FROM USER clauses. The FROM [FILE] (IN [FILE]) and FROM USER clauses are mutually exclusive in an ADASEL run. An ADASEL run should either process protection log data by file (FROM [FILE] or IN [FILE] clauses) or by user (FROM USER clause), but not both. Any attempt to process protection log data by file and user in the same ADASEL run will cause errors.

The FROM [FILE] and IN [FILE] clauses are equivalent clauses. Both specify the number of the Adabas file for which protection log data will be selected and processed in the ADASEL run. In both cases the keyword FILE is optional. Valid file numbers range from 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.

FROM USER Clause

One of the FROM [FILE], IN [FILE} or the FROM USER clauses is required in a SELECT parameter. Only one is required; if more than one is specified, errors will result. The FROM and IN [FILE] clauses are described in *FROM and IN [FILE]* .

Note:
A single ADASEL run cannot include both FROM [FILE] (or IN [FILE]) clauses and FROM USER clauses. The FROM [FILE] (IN [FILE]) and FROM USER clauses are mutually exclusive in an ADASEL run. An ADASEL run should either process protection log data by file (FROM [FILE] or IN [FILE] clauses) or by user (FROM USER clause), but not both. Any attempt to process protection log data by file and user in the same ADASEL run will cause errors.

The FROM USER clause indicates that all records from all files that satisfy the selection criteria should be selected and processed in the ADASEL run, regardless of file number.

To select records from a specific user or terminal ID, specify the actual user or terminal ID used for selection using the WITH clause of the SELECT parameter. For example, the following SELECT parameter selects all protection log data for user or terminal ID ETID1:

```
SELECT ALL FROM USER FDTINPUT
   WITH USERID='ETID1'
   DISPLAY ALL
END
```

Be aware that FROM USER clauses can generate quite a bit of output for display, so be prepared to limit the selection criteria in the SELECT parameter if necessary. In addition, you can use the NU global parameter to limit the number of user or terminal IDs that can be processed by a SELECT parameter. If this number is exceeded, errors result (the default is 20). For more information about the NU global parameter, read *SET GLOBALS Parameter*.

Finally, when FROM USER is specified, records from different files may be encountered for the same SELECT statement. For this reason, no field names can be specified in any associated value criteria or in DISPLAY instructions. However, the DISPLAY ALL option can be used to display all fields of a selected record, regardless of the file the record belongs to.

FDTINPUT Parameter

The optional FDTINPUT parameter can be used to indicate that a different FDT from the FDT on the database should be used for specific files or users selected in the ADASEL run. This functionality allows you to handle data situations where the FDT has been modified in some way so it differs from the actual data in the database. In these cases, an older FDT might be necessary for ADASEL to accurately process the data in the database. For more information, read *FDTINPUT Parameter*.

STARTING FROM and ENDING AT *date-time* Clauses

The optional STARTING FROM and ENDING AT clauses can be used to restrict selections to records added, updated, or deleted within a time range. 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 optional WITH clause can be 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, *all* specified conditions 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 can be used to select records and execute output instructions on a conditional basis. An IF statement is optional in a SELECT parameter, but can be used to specify conditional output instructions for the SELECT parameter. At least one output instruction is required in a SELECT parameter, so if one has not been specified outside an IF statement, an IF statement is necessary to supply the output instruction information.

By default, ADASEL permits up to 20 nested IF statements. To change this, use the NIF ADASEL global parameter. For further information about the NIF global parameter, read *SET GLOBALS Parameter*.

```
IF { value-criterion        } [ AND  { value-criterion        } ]
   { field-name  CHANGES    } [ OR   { field-name  CHANGES    } ]

   [THEN] { output-instruction                    } [ ELSE { output-instruction                    } ]
          { IF-statement                          } [      { IF-statement                          } ]
          { DO { output-instruction }... DOEND    } [      { DO { output-instruction }... DOEND    } ]
          { IGNORE                                } [      { IGNORE                                } ]
```

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

The "*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:
By default, 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, either specify the BOTH option in the DISPLAY instruction or specify the LOGINFO, EXTENDED, or SPANREC options on the OUTPUT instruction for the run. For more information, read *DISPLAY Instruction* or *OUTPUT Instruction*.

The syntax for DO group is as follows:

```
DO { output-instruction }... DOEND
```

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.

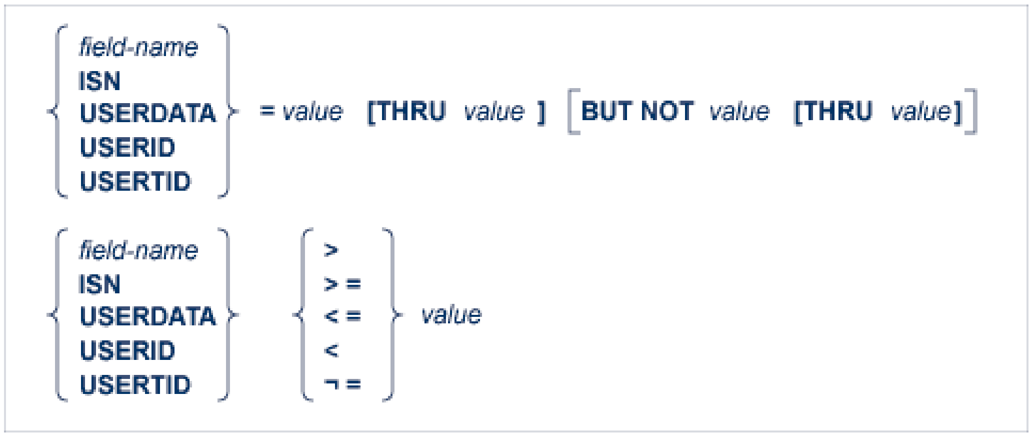
IGNORE instructs ADASEL not to display or output an item. Neither the before-image (BI) or the after-image (AI) is produced as output when an item is ignored. When specified in a THEN instruction, IGNORE will not display or output the item *if it meets* the specified value-criterion or the CHANGES criterion of the IF statement. When specified in an ELSE instruction, IGNORE will not display or output the item *if it does not meet* the specified value-criterion or the CHANGES criterion of the IF statement.

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 within the select records.

Object of the Comparison

ADASEL can compare a value or range of values to any of the following:

- 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 .

- The 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

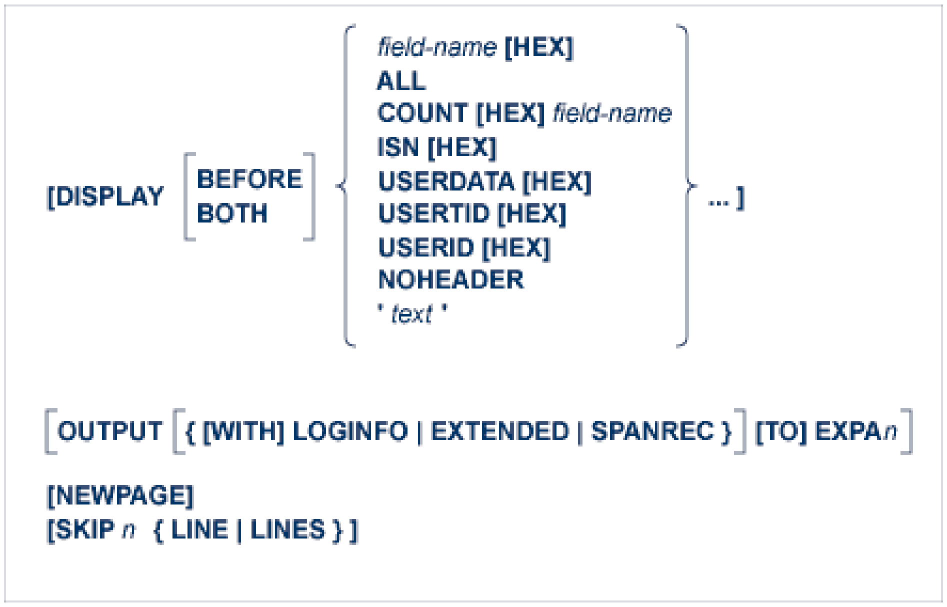
Example:

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

ADASEL output instructions include DISPLAY, OUTPUT, SKIP, and NEWPAGE. At least one output instruction is required, either separately or within an IF statement. Multiple output instructions can be specified. The syntax is shown below:



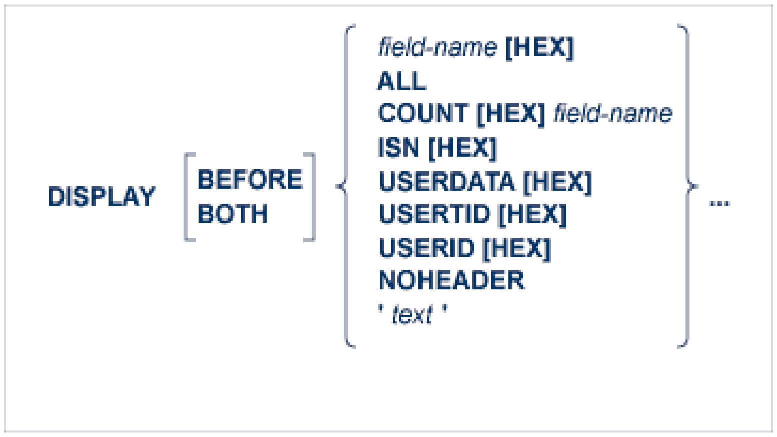
```
                              ⎧ field-name [HEX]          ⎫
                              ⎪ ALL                       ⎪
                              ⎪ COUNT [HEX] field-name    ⎪
                              ⎪ ISN [HEX]                 ⎪
[DISPLAY  ⎡ BEFORE ⎤          ⎨ USERDATA [HEX]            ⎬ ... ]
          ⎣ BOTH   ⎦          ⎪ USERTID [HEX]             ⎪
                              ⎪ USERID [HEX]              ⎪
                              ⎪ NOHEADER                  ⎪
                              ⎩ ' text '                  ⎭

[OUTPUT [{ [WITH] LOGINFO | EXTENDED | SPANREC }] [TO] EXPAn]

[NEWPAGE]
[SKIP n  { LINE | LINES } ]
```

This section describes each of these output instructions.

- DISPLAY Instruction
- OUTPUT Instruction
- NEWPAGE Instruction
- SKIP Instruction

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.
ALL	displays all fields of each selected record, including MU and PE group fields and their occurrence counts. For an FDT with many fields, you may need to increase the LST global parameter in a SET GLOBALS statement to provide sufficient space for the large number of records. This option is available for both SELECT FROM USER and SELECT FROM FILE (IN FILE) clauses in a SELECT parameter.
BEFORE	indicates that only before image of protection log data should be output. This option is valid only within IF statements containing one or more CHANGES specifications. This option is optional and is mutually exclusive with the BOTH option. If neither BOTH nor BEFORE is specified, only the after image of protection log data is output.

BOTH	indicates that both before and after images of protection log data should be output. This option is valid only within IF statements containing one or more CHANGES specifications. This option is optional and is mutually exclusive with the BEFORE option. f neither BOTH nor BEFORE is specified, only the after image of protection log data is output.
COUNT	displays the number of occurrences in the selected data of the MU field or PE group field specified after the COUNT option. If one or more values of the MU or PE field are to be displayed as well, they must be specified separately.
option	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.
ISNv	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
```

Select records that have been modified and display the occurrence count, followed by the hexadecimal values of the seventh through twelfth occurrences of the MU field XX:

```
SELECT UPDATED RECORDS FROM FILE 32
   DISPLAY COUNT XX XX7-12 HEX
END
```

Select records that have been modified and display the occurrence count for PE group field XX, followed by the numbers of the YY values in the first through last XX PE group occurrence, followed by all YY values in each XX PE group occurrence (in this example YY is an MU field within the XX PE group):

```
SELECT UPDATED RECORDS FROM FILE 32
   DISPLAY COUNT XX COUNT YY1-N YY1-N(1-N)
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
```

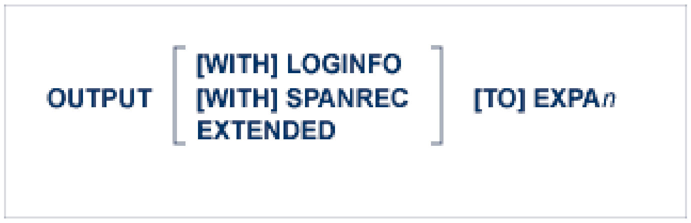
```
SELECT ALL FROM FILE 12
   DISPLAY JT1-5 SA1-5(1-N) BO1-5
END
```

```
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.



```
         [ [WITH] LOGINFO ]
OUTPUT   [ [WITH] SPANREC ]   [TO] EXPAn
         [ EXTENDED       ]
```

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 SPANREC, LOGINFO, or EXTENDED parameter is specified. LOGINFO and EXTENDED are used to display additional information. The SPANREC parameter indicates that alternate headers should be used to handle spanned records.

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.

SPANREC

When SPANREC is specified, the new spanned record SELH and SELC output headers are used for all EXPA*n* output. DSECTs for the SELH and SELC headers can be found in the Adabas source library. These new spanned record headers are used when any decompressed logical record from the PLOG exceeds the EXPA*n* physical record limitation. In this case, the SELH header will prefix every logical record written to EXPA*n*; subsequent physical records belonging to the same logical record will be prefixed by the SELC header.

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 Instruction

The NEWPAGE instruction and SKIP instructions control page formatting. The NEWPAGE instruction forces a page eject before displaying the next line of data. In the following example, a page eject occurs every time the value of the BA field changes:

```
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
END
```

SKIP Instruction

The NEWPAGE instruction and SKIP instructions control page formatting. The SKIP instruction prints the specified number of blank lines before displaying the next line of data. In the following example, two blank lines are printed every time the value of the BA field changes.

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

The END Keyword

The END keyword is a required element of the SELECT parameter and must be specified last.

Examples

This section provides several examples of SELECT parameters.

The following SELECT parameter will output to data set DD/EXPA1 all new data records inserted by user ETID1 for November 1st, 2008:

```
SELECT NEW RECORDS FROM USER
    STARTING FROM 20081101/000000
    ENDING AT 20081101/240000
    WITH USERID=’ETID1’
    OUTPUT TO EXPA1
END
```

The following SELECT parameter will output to data set DD/EXPA2 the after images of all data records updated by any users working from terminals CICS1000 through CICS9999 prior to the end of November 1st, 2008. The output will include extended LOGINFO data:

```
SET GLOBALS NU=50
SELECT NEW RECORDS FROM USER
    ENDING AT 20081101/240000
    WITH USERTID EQ ’CICS1000’ THRU ’CICS9999’
    OUTPUT EXTENDED TO EXPA2
END
```

The following SELECT parameter will display fields AA and AB of all data records from file 200 that were inserted, updated, or deleted by user ETID1:

```
SELECT ALL RECORDS FROM FILE 200
    WITH USERID EQ ’ETID1’
    DISPLAY AA AB
END
```

Finally, the following SELECT parameter will display user IDs for all users who updated the data base:

```
SELECT UPDATED RECORDS FROM USER
    DISPLAY USERID
END
```