

READ WORK FILE

Structured Mode Syntax

```

READ WORK [FILE] work-file-number [ONCE]
  {
    RECORD operand1
    { [AND] [SELECT] { [OFFSET n]
                      [FILLER nX]... operand2 } ... }
    [GIVING LENGTH operand3]
    { AT [END] [OF] [FILE]
      statement...
      END-ENDFILE
    }
    statement...
  }
END-WORK
    
```

Reporting Mode Syntax

```

READ WORK [FILE] work-file-number [ONCE]
  {
    RECORD {operand1 [FILLER nX]}...
    { [AND] [SELECT] { [OFFSET n]
                      [FILLER nX]... operand2 } ... }
    [GIVING LENGTH operand3]
    { AT [END] [OF] [FILE] { statement
                          [DO statement...DOEND]
                        }
    }
    statement...
  }
[LOOP]
    
```

Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
Operand1	S A G	A N P I F B D T L C G	yes	yes
Operand2	S A G	A N P I F B D T L C	yes	yes
Operand3	S	I	yes	yes

Note:

Format C is not valid for Natural Connection.

Related Statements: DEFINE WORK FILE | CLOSE WORK FILE

Function

The READ WORK FILE statement is used to read data from a non-Adabas physical sequential work file. The data is read sequentially from the work file. How it is read is independent of how it was written to the work file.

On mainframe computers, this statement can only be used within a program to be executed under Complete, CMS, TSO or TIAM, or in batch mode. The appropriate JCL must be supplied in the execution JCL when a work file is to be read. See the Natural Operations for Mainframes documentation for additional information.

READ WORK FILE initiates and executes a processing loop for reading of all records on the work file. Automatic break processing may be performed within a READ WORK FILE loop.

Note:

When an end-of-file condition occurs during the execution of a READ WORK FILE statement, Natural automatically closes the work file.

Note for Entire Connection:

If an Entire Connection work file is read, no I/O statement may be placed within the READ WORK FILE processing loop.

Note for Open Systems:

If an ASCII work file is read, it is possible that an empty record is returned as the last record after the last physical record. This is caused by the fact that Natural does not read individual records, but reads larger blocks of the work file in order to optimize file-access performance.

work-file-number

The number of the work file (as defined to Natural) to be read.

ONCE Option

ONCE is used to indicate that only one record is to be read. No processing loop is initiated (and therefore the loop-closing keyword END-WORK or LOOP must not be specified). If ONCE is specified, the AT END OF FILE clause should also be used.

If a READ WORK FILE statement specified with the ONCE option is controlled by a user-initiated processing loop, an end-of-file condition may be detected on the work file before the loop ends. All fields read from the work file still contain the values from the last record read. The work file is then repositioned to the first record which will be read upon the next execution of READ WORK FILE ONCE.

Variable Index Range

When reading an array from a work file, you can specify a variable index range for the array. For example:

READ WORK FILE work-file-number #ARRAY (I:J)

RECORD Option

If RECORD is specified, all fields in each record read are made available for processing. An operand list (*operand1*) must be provided corresponding to the layout of the record. A FILLER *nX* entry indicates *n* bytes are to be skipped in the input record. The record as defined in the RECORD clause must be in contiguous storage. FILLER is not permitted in structured mode.

In structured mode, or if the record to be used is defined using a DEFINE DATA statement, only one field (or group) may be used. FILLER is not permitted in this case.

No checking is performed by Natural on the data contained in the record. It is the user's responsibility to describe the record layout correctly in order to avoid program abends caused by non-numeric data in numeric fields. Because no checking is performed by Natural, this option is the fastest way to process records from a sequential file. The record area defined by operand1 is filled with blanks before the record is read. Thus, an end-of-file condition will return a cleared area. Short records will have blanks appended.

Note for Entire Connection:

If an Entire Connection work file is read, the RECORD option cannot be used.

SELECT Option - default

If SELECT is specified, only those fields specified in the operand list (*operand2*) will be made available. The position of the field in the input record may be indicated with an OFFSET and/or FILLER specification. OFFSET 0 indicates the first byte of the record. FILLER *nX* indicates that *n* bytes are to be skipped in the input record. On mainframes, OFFSET cannot be specified for work files defined as TYPE UNFORMATTED.

Natural will assign the selected values to the individual fields and check that numeric fields as selected from the record actually contain valid numeric data according to their definition. Because checking of selected fields is performed by Natural, this option results in more overhead for the processing of a sequential file.

If a record does not fill all fields specified in the SELECT option, the following applies:

- For a field which is only partially filled, the section which has not been filled is reset to blanks or zeros.
- Fields which are not filled at all still have the contents they had before.

Note:

If the file types ENTIRECONNECTION or TRANSFER are read, the OFFSET option is ignored.

Field Lengths

The field lengths in the operand list are determined as follows:

- For A, B, I and F fields, the number of bytes in the input record is the same as the internal length definition.
- For N format fields, the number of bytes in the input record is the sum of internal positions before and after the decimal point. The decimal point and sign do not occupy a byte position in the input record.
- For P, D and T fields, the number of bytes in the input record is the sum of positions before and after the decimal point plus 1 for the sign, divided by 2 rounded upwards.
- For L format fields, 1 byte is used. For C format fields, 2 bytes are used.

Examples of Field Lengths:

Field Definition	Input Record
#FIELD1 (A10)	10 bytes
#FIELD2 (B15)	15 bytes
#FIELD3 (N1.3)	4 bytes
#FIELD4 (N0.7)	7 bytes
#FIELD5 (P1.2)	2 bytes
#FIELD6 (P6.0)	4 bytes

See also the section Definition of Format and Length.

GIVING LENGTH operand3

The GIVING LENGTH clause can be used to retrieve the actual length of the record being read. The length (number of bytes) is returned in *operand3*. *Operand3* must be defined with format/length I4.

If the work file is defined as TYPE UNFORMATTED, the length returned indicates the number of bytes read from the byte-stream, including bytes skipped using the FILLER operand.

AT END OF FILE

The AT END OF FILE clause can only be used in conjunction with the ONCE option. If the ONCE option is used, this clause should be specified to indicate the action to be taken when an end-of-file condition is detected.

If the ONCE option is not used, an end-of-file condition is handled like a normal processing loop termination.

Handling of Large and Dynamic Variables

When the operand list includes a dynamic variable (that could change in size, depending on the WRITE WORK FILE statement), the VARIABLE entry must be specified in all WRITE WORK FILE statements.

ASCII, ASCII-COMPRESSED and SAG (binary)

The work file types ASCII, ASCII-COMPRESSED and SAG (binary) cannot handle dynamic variables and will produce an error. They can, however, handle large variables with a maximum field/record length of 32766 bytes.

TRANSFER and ENTIRE CONNECTION

The work file type TRANSFER can handle dynamic variables. There is no size limit here. The work file type ENTIRE CONNECTION cannot handle dynamic variables. They can both, however, handle large variables with a maximum field/record length of 107341824 bytes.

The RECORD option is not allowed if any dynamic variables are used.

PORTABLE and UNFORMATTED

Large and Dynamic Variables can be written into work files or read from work files using the two work file types PORTABLE and UNFORMATTED. For these types, there is no size restriction for dynamic variables. However, large variables may not exceed a maximum field/record length of 32766 bytes.

Reading a dynamic variable from a PORTABLE work file leads to resizing to the stored length.

Reading a dynamic variable from an UNFORMATTED work file puts the complete rest of the file into the variable (from the current position). If the file exceeds 1073741824 bytes, then a maximum 1073741824 bytes is placed into the variable. Reading a dynamic variable from a FORMATTED work file fills the variable in its currently defined length (including length 0). If the end-of-file is reached, the remainder of the current field is filled with blanks. The subsequent fields are unchanged.

Example

```

/* EXAMPLE 'RWFEX1': READ WORK FILE
/*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
1 #RECORD
  2 #PERS-ID (A8)
  2 #NAME (A20)
END-DEFINE
/*****
FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
WRITE WORK FILE 1
  PERSONNEL-ID
  NAME
END-FIND
/*****
/* ...
/*****
READ WORK FILE 1
  RECORD
  #RECORD
  DISPLAY NOTITLE #PERS-ID #NAME
END-WORK
/*****
END

```

PAGE 1

87-03-27 15:46:58

#PERS-ID	#NAME
-----	-----
11100328	BERGHAUS
11100329	BARTHEL
11300313	AECKERLE
11300316	KANTE
11500304	KLUGE
11500308	DIETRICH
11500318	GASSNER
11500343	ROEHM
11600303	BERGER
11600320	BLAETTEL
11500336	JASPER
11100330	BUSH
11500328	EGGERT