

WRITE WORK FILE

WRITE WORK [FILE] <i>work-file-number</i> [VARIABLE] <i>operand1</i> ...

Operand	Possible Structure				Possible Formats								Referencing Permitted	Dynamic Definition			
Operand1	C	S	A	G	A	N	P	I	F	B	D	T	L	C	G	yes	no

Note:

Neither Format C nor Format G is valid for Natural Connection.

Related Statements: [DEFINE WORK FILE](#) | [READ WORK FILE](#) | [CLOSE WORK FILE](#) | [DOWNLOAD PC FILE](#)

Function

The WRITE WORK FILE statement is used to write records to a physical sequential work file.

On mainframe computers, this statement can only be used in batch mode, or under Com-plete, CMS, TSO and TIAM.

It is possible to create a work file in one program or processing loop and to read the same file in a subsequent independent processing loop or in a subsequent program using the READ WORK FILE statement.

work-file-number

The work file number (as defined to Natural) to be used.

VARIABLE

It is possible to write records with different fields to the same work file with different WRITE WORK FILE statements. In this case, the VARIABLE entry must be specified in all WRITE WORK FILE statements. The records on the external file will be written in variable format. Natural will write all output files as variable-blocked (unless you specify a record format and block size in the execution JCL).

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

Fields - operand1

With *operand1* you specify the fields to be written to the work file. These fields may be database fields, user-defined variables, and/or fields read from another work file using the READ WORK FILE statement.

A database array may be referenced with one single range of indices which indicates the occurrences that are to be written to the work file. Groups from database files may be referenced using the group name. All fields belonging to that group will be written to the work file individually.

Variable Index Range

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

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

External Representation of Fields

Fields written with a WRITE WORK FILE statement are represented in the external file according to their internal definition. No editing is performed on the field values.

For fields of format A and B, the number of bytes in the external file is the same as the internal length definition as defined in the Natural program. No editing is performed and a decimal point is not represented in the value.

For fields of format N, the number of bytes on the external file is the sum of internal positions before and after the decimal point. The decimal point is not represented on the external file.

For fields of format P, the number of bytes on the external file is the sum of positions before and after the decimal point, plus 1 for the sign, divided by 2, rounded upward to a full byte.

Note:

No format conversion is performed for fields that are written to a work file.

Examples of Field Representation:

Field Definition	Output 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

Note:

When the system functions AVER, NAVER, SUM or TOTAL for numeric fields (format N or P) are written to a work file, the internal length of these fields is increased by one digit (for example, SUM of a field of format P3 is increased to P4). This has to be taken into consideration when reading the work file.

Handling of large and dynamic variables

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 with a maximum field/record length of 32766 bytes. The work file type ENTIRE CONNECTION cannot handle dynamic variables. They can both, however, handle large variables with a maximum field/record length of 1073741824 bytes.

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.

For the work file type PORTABLE, the field information is stored within the work file. The dynamic variables are resized during READ if the field size in the record is different from the current size.

In the WRITE WORK statement, fields are written to the file specified with their byte length. All data types (DYNAMIC or not) are treated the same. No structural information is inserted. Note that Natural uses a buffering mechanism, so you can expect the data to be completely written only after a CLOSE WORK. This is especially important if the file is to be processed with another utility while Natural is running.

With the READ WORK statement, fields of fixed length are read with their whole length. If the end-of-file is reached, the remainder of the current field is filled with blanks. The following fields are unchanged. In the case of DYNAMIC data types, all the remainder of the file is read unless it exceeds 1073741824 bytes. If the end of file is reached, the remaining fields (variables) are kept unchanged (normal Natural behavior).

On mainframe computers, work file type UNFORMATTED can be used to write variables, the size of which exceeds the maximum record length. See also the section Work File Access With Large and Dynamic Variables - Mainframes.

Example

```

/* EXAMPLE 'WWFEX1': WRITE WORK FILE
/*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
END-DEFINE
/*****
FIND EMPLOY-VIEW WITH CITY = 'LONDON'
WRITE WORK FILE 1
  PERSONNEL-ID
  NAME
END-FIND
/*****
END

```