

Processing Work Files and Nested Loops

This section describes restrictions on the use of work file attributes, the support of work file formats and the impact of READ loops.

- Work File Attributes
 - Streaming
 - Dynamic Variables in READ WORKFILE
 - Nested READ Loops
 - Subsequent READ Loops
-

Work File Format and Attributes

Below are the restrictions that apply to the use of work file attributes:

- Accessing PC work files is restricted to a record length of 32767 bytes.
- Natural Connections does not support work files of type UNFORMATTED.
A work file is always transferred in formatted mode and contains record-oriented data only. For a work file of the type UNFORMATTED, at FILE OPEN, Natural issues the error message NAT1190. To transfer byte-streamed data, see Streaming below.

Streaming

Entire Connection provides the option to transfer byte-streamed data that are non-record-oriented. A byte-streamed data transfer is activated when a READ WORKFILE or WRITE WORKFILE statement is coded with only one single operand of binary format.

Below is information on:

- Downloading and Uploading Binary Data

Downloading and Uploading Binary Data

Binary data is usually object code or executable code that does not contain displayable or printable characters. To prevent standard character translations being performed during data transfer, Natural and Entire Connection use special methods for transferring binary data.

To download binary data

- Define a binary variable (B1 to B32767).
- If the last block of downloaded data contains less data than the block size chosen, insert X'FF' at the position that marks the end of the binary data. (If you omit X'FF', the rest of the last block will be filled with X00.)

To upload binary data

- Define a binary variable (B1 to B32767).
- Remove X'FF from the last block. X'FF marks the end of the binary data.

Dynamic Variables in READ WORKFILE

If you define a dynamic variable of the format binary or alphanumeric as operand of a READ WORKFILE statement, when processing the corresponding READ loop, any resize operation on this variable will only be valid until the next READ is performed. While processing the READ, Natural resizes all dynamic variables to the size they had at OPEN time. This is required in the OPEN process which determines the record layout. The record layout is mandatory for processing the corresponding work file. The record layout is valid until the next CLOSE operation occurs.

Exception:

An internal resize cannot be performed for inner loops if nested READ loops are processed on the same work file. See also the programming recommendations about nested loops below. If a dynamic variable of Size 0 is used as the only operand of a READ WORKFILE statement, Natural issues the OPEN error NAT1500.

Nested READ Loops

Do not specify nested READ loops on one work file. The result of the inner loop(s) can be unpredictable if the the operands of the inner loop do not correspond with the operands of the outer loop. The reason is, that all records uploaded from the PC are processed in the format that was determined during the OPEN of the outermost loop.

Below are example programs that demonstrate the unpredictable results the inner loop(s) of nested READ loops can have:

- Example of Inner READ Loop
- Example of Read Loop and CALLNAT

Example of Inner READ Loop

In the example program PCNESTED, during READ processing, another READ is performed:

```

/* PCNESTED
/*
DEFINE DATA LOCAL
  1 #REC1   (A) DYNAMIC
  1 #NUMBER (N10)
END-DEFINE
*
MOVE ALL 'TEST RECORD 1' TO #REC1 UNTIL 100
READ WORK FILE 1 #REC1
  READ WORK FILE 1 #NUMBER
  DISPLAY #NUMBER
END-WORK
END-WORK
END

```

Example of Read Loop and CALLNAT

In the example program PCMAIN and subprogram PCRSUB01, during READ loop processing, an external object is called:

```

/* PCMAIN
/*
DEFINE DATA
LOCAL
  1 RECL (A2000)
  1 REDEFINE RECL
    2 RECNR (N4)
  1 CO (N4)
END-DEFINE
*
WRITE WORK 1 COMMAND
'SET PCFILE 2 UP DATA C:/TSTPCAM/PCMAIN.TXT'
READ WORK 2 RECL
  DISPLAY RECL (AL=72)
  CALLNAT 'PCRSUB01' RECL
END-WORK
END

```

Subprogram PCRSUB01

```

/*Subprogram PCRSUB01
/*
DEFINE DATA
PARAMETER
  1 RECL (A2000)
LOCAL
  1 #CC1 (A20)
  1 #CC2 (N4)
*
END-DEFINE
READ WORK 2 RECL
  #CC1 #CC2
  DISPLAY #CC1 #CC2
END-WORK
END

```

Subsequent READ Loops

If a READ loop is terminated by a conditional ESCAPE, close the work file explicitly with the CLOSE WORKFILE statement so that the same work file can be processed in a subsequent READ in the same object.

Exception:

You can omit the CLOSE WORKFILE if you need not read the file again from the beginning, and if the subsequent READ uses the same record layout as the preceding one.

Below is an example that demonstrates how to correctly code a program with two READ loops on one work file:

- Example of Loop with ESCAPE and CLOSE

Example of Loop with ESCAPE and CLOSE

In the example program PCESCAPE, the work file is explicitly closed after the first READ loop has been terminated by ESCAPE BOTTOM so that the second READ loop must reopen the work file:

```
/*PCESCAPE
/*
DEFINE DATA
LOCAL
  1 #CC1      (A20)
  1 #CC2      (A40)
  1 #COUNTER  (I2)
*
END-DEFINE
READ WORK 2 #CC1
  DISPLAY #CC2
  ADD 1 TO #COUNTER
  IF #COUNTER GE 3
    ESCAPE BOTTOM
  END-IF
END-WORK
CLOSE WORK FILE 2
*
READ WORK 2 #CC2
  DISPLAY #CC2
END-WORK
END
```