Processing Work Files and Nested Loops

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

- Work File Format and Attributes
- Maximum File Transfer Record Length for Natural Connection
- Streaming
- Dynamic Variables in READ WORK FILE
- Nested READ Loops
- Subsequent READ Loops
- Buffer Allocation for Large Upload Records

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 fixed record length of 1073741823 bytes or 32767 bytes when using the statement WRITE WORK FILE VARIABLE. Depending on the Entire Connection version installed on the PC, additional restrictions may apply as described below.
- Natural Connection does not support work files of the type UNFORMATTED. A work file is always transferred in formatted mode and contains record-oriented data only. When a work file of the type UNFORMATTED is opened, Natural Connection switches to the type FORMATTED and executes any WRITE WORK FILE statement with the option VARIABLE. To transfer byte-streamed data, see *Streaming* below.

Maximum File Transfer Record Length for Natural Connection

As of Natural Connection Version 4.1, the maximum record length supported for file transfer depends on the version of Entire Connection installed on the PC.

For Entire Connection up to Version 4.2, the maximum record length is limited by the number of bytes that can be displayed on the appropriate 3270 model. For example, for a 3270 Model 2 device the record length is 24*80 = 1920 bytes. Since all data buffers are enclosed by a header and trailer, the resulting net record length is 1887 bytes.

For Entire Connection Version 4.3.1, the maximum record length is limited to 32 KB - 1 byte = 32767 bytes.

As of Entire Connection Version 4.3.2 Patch Level 1 and Entire Screen Builder Version 5.2.1, the maximum record length is increased to 1 GB - 1 byte = 1073741823 bytes. But writing work files in variable format (`WRITE WORK VARIABLE`) is still restricted to a maximum record length of 32 KB - 1 byte.

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 WORK FILE` or `WRITE WORK FILE` statement is coded with only one single operand of binary format.

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

1. Define a binary variable.
2. 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

1. Define a binary variable.
2. Remove `X'FF` from the last block. `X'FF` marks the end of the binary data.

Dynamic Variables in READ WORK FILE

If you define a dynamic variable of the format binary or alphanumeric as operand of a `READ WORK FILE` 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 when the work file was opened. 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 of the work file 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 WORK FILE` statement, Natural issues the 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 operands of the inner loop do not correspond to the operands of the outer loop. The reason is that all records uploaded from the PC are processed in the format that was determined when the work file was

opened in 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 WORK FILE statement so that the same work file can be processed in a subsequent READ in the same object.

Exception: You can omit the CLOSE WORK FILE 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

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
```

Buffer Allocation for Large Upload Records

If Natural Connection uploads a record that is larger than one physical block, Natural Connection collects all blocks that belong to the record in the appropriate work file area. The record will then be decompressed and passed to the Natural data area.

The total space allocated by all temporary buffers is up to 3 times the size of the record to be uploaded.

Example Statement

```
READ WORK FILE 1 #var
```

where 1 is the number of the work file and `#var` a variable of the format B 10000. In this case, the temporary Natural work area requires approximately 30000 bytes.