

# 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 nested READ loops.

- Work File Attributes
- Nested READ Loops
- Examples of Nested 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 32 K.
- Natural Connections does not support the Natural statement DEFINE WORKFILE TYPE 'UNFORMATTED'. A work file is always transferred in formatted mode and contains **non-streamed data (???)** only, regardless of whether the work file has been specified as FORMATTED or UNFORMATTED. For the work file type UNFORMATTED, at FILE OPEN, Natural Connection (**oder Natural???)** issues the error NAT1190. To transfer work files with non-streamed (unformatted???) data, see Streaming below.

Below is information on:

- Streaming

### Streaming

Entire Connection provides the **streaming mode (???)**, which is a non-record-oriented mode where data is not formatted. (???) Streaming mode processing is activated when a READ WORKFILE or WRITE WORKFILE statement is coded with only one binary operand.

## Nested READ Loops

Do not code nested READ loops into **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.

**???)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.???)Absatz verschoben - o.k.???)**

Below is information on:

- Dynamic Variables in READ WORKFILE

### 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 before 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](#). ??? If a dynamic variable of Size 0 is used as the only operand of a READ WORKFILE statement, Natural Connection ([oder Natural???](#)) issues the OPEN error NAT1500.

## Examples of Nested READ Loops

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 Inner CALLNAT Loop
- Example of Loop with CLOSE and ESCAPE

### 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 Inner CALLNAT Loop

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)
*

```

### Example of Loop with CLOSE and ESCAPE

In the example program PCESCAPE, the work file is explicitly closed and a subsequent READ loop is separated from the previous one by using ESCAPE BOTTOM:

```

/*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

```

