Work Files

Work files are files to which data can be written and from which data can be read by Natural programs. They are used for intermediate storage of data and for data exchange between programs. Data can be transferred from or to a work file by using the Natural statements `READ WORK FILE` and `WRITE WORK FILE`.

This chapter covers the following topics:

- Defining Work Files
- Work File Formats
- Special Considerations for Work Files with the Extension NCD

Defining Work Files

Using the Configuration Utility or the `DEFINE WORK FILE` statement, you can assign names (including the path) for up to 32 work files.

The maximum number of work files that can be used depends on the setting of the parameter `WORK`.

If you run a program which uses a work file for which a name and path has not been assigned, Natural automatically creates the file name and writes the work file into the temporary directory specified in the local configuration file. The name of such a file consists of the specified work file number and an arbitrary number assigned by the operating system. The generation of the work file name is based on an algorithm which tries to generate a unique name. Depending on the Natural parameter `TMPSORTUNIQ`, the naming convention may vary. If work file names are referenced from outside Natural, it is recommended that you specify the names explicitly to avoid problems identifying the files.

The following topics are covered below:

- Defining Work File Names with the Configuration Utility
- Defining Work File Names with Environment Variables
- Defining Work File Names with an Application Programming Interface

Defining Work File Names with the Configuration Utility

In the Configuration Utility, the work file names are assigned in the category **Work Files** of a parameter file. The above mentioned parameters `WORK` and `TMPSORTUNIQ` can also be found in this category. See *Work File Assignments* in the *Configuration Utility* documentation for further information.

Tip:
Locate the work file assignments by searching for "Work Files". See *Finding a Parameter* in the *Configuration Utility* documentation for further information.

Defining Work File Names with Environment Variables

The following topics are covered below:

- General Information
- Delimiters of Environment Variables
- Dollar Sign ($) in the File Name

General Information

Work files can also be defined by using Windows environment variables. Once you have defined the work file names in the parameter file, the work file names can be set without further change to the parameter file. For example, when you specify the following name for a work file in the parameter file (or in a `DEFINE WORK FILE` statement):

```
%Natural%\%myfile%
```

and assume the following settings in your operating system:

```
set Natural=D:\natural
set myfile=sub\test
```

this will expand into the following file name:

```
D:\natural\sub\test
```

Delimiters of Environment Variables

Names of environment variables are delimited by special characters. A left-hand delimiter is to the left of a variable, a right-hand delimiter is to the right.

For example, the string `%TEMP%` identifies an environment variable named `TEMP`; `%` is used as both the left-hand and right-hand delimiter.

Valid delimiters are:

Type of Delimiter	Valid Delimiters
Left-hand delimiter	% $
Right-hand delimiter	% / . \

Note:
The end-of-string mark is by default a right-hand delimiter, i.e. `%TEMP` is recognized as an environment variable named `TEMP`.

Although "%" is the only valid left-hand delimiter for environment variables in Windows, Natural for Windows allows "%" and "$" as left-hand delimiters in order to preserve upward compatibility with previous versions. This setting allows UNIX-like work file name assignments in a Windows session. `$TEMP` is recognized in Natural for UNIX as well as in Natural for Windows as the environment variable

TEMP.

Example:

The following lines of Natural code are interpreted as being the same:

```
DEFINE WORK FILE 1 '$TEMP\myfile.dat'
```

and

```
DEFINE WORK FILE 1 '%TEMP%\myfile.dat'
```

TEMP is recognized as an environment variable. The string $TEMP (or %TEMP%) is replaced at runtime by the contents of the environment variable TEMP.

Dollar Sign ($) in the File Name

A dollar sign ($) in a file name has two meanings:

- If "$" appears on the left or in the middle of a string embedded in delimiters, it will be interpreted as the left-hand delimiter of the environment variable being used. All characters following the left-hand delimiter up to the right-hand delimiter or EOS are considered to be the name of an environment variable.
- If "$" is the last character of a string, it is not considered to be a delimiter character. It is a part of the string scanned.

Example:

The following line of Natural code does not result in an error:

```
DEFINE WORK FILE 1 '\\MYPC\C$\myfile.dat'
```

\\MYPC\C$ is considered to be a default share. "C$" is a valid directory.

However, the following line of Natural code may result in an error, depending on whether A has been defined or not:

```
DEFINE WORK FILE 1 '\\MYPC\C$A\myfile.dat'
```

"A" is interpreted as an environment variable since it is preceded by a dollar sign. If "A" has not been defined, an error will occur. If "A" has been defined, an error does not occur.

Defining Work File Names with an Application Programming Interface

You can also define work files with the application programming interface USR1050N in library SYSEXT.

Work File Formats

The format of a work file depends on the work file type that has been defined. Different work file formats are available. Natural recognizes the format by checking the file name and its extension:

```
file-name.extension
```

where *file-name* can have a maximum of 8 characters and *extension* can have a maximum of 3 characters.

The work file formats are:

- Binary Format
- ASCII Format
- Entire Connection Format
- Portable Format
- Unformatted Format
- CSV Format

See also *Work Files and Print Files* in the *Unicode and Code Page Support* documentation.

Binary Format

Possible type: SAG.

This format, which is specific to Software AG, is the preferred format since it can be used with all data types.

Each record that is written is preceded by two bytes which contain the length of the record.

To define binary format for a work file, use either a file name with a period and the extension "SAG" (for example, <*file-name*>.*SAG*), or just the file name without a period (for example, <*file-name*>).

ASCII Format

Possible types: ASCII and ASCII compressed.

Since each written record is terminated with a carriage return and line feed (CR/LF), ASCII format is only recommended for alphanumeric data.

To define ASCII format for a work file, enter either a file name with a period and any extension except "SAG" and "NCD" (for example, <*file-name*>.<*ext*>), or a file name with a period and without an extension (for example, <*file-name*>).

Entire Connection Format

Possible type: Entire Connection.

The product Entire Connection uses two files: a data file which contains the actual data and a format file which contains formatting information about the data in the data file.

Natural automatically generates the corresponding format file for the type Entire Connection. The format file has the same name as the data file, however the extension is "NCF". For detailed information on the content of a format file with the extension "NCF", see the Entire Connection documentation.

To define Entire Connection format for a work file, enter a file name with a period and the extension "NCD" (for example, *<file-name>.NCD*).

You can read/write work files in Entire Connection format directly from/to your local disk.

See also *Special Considerations for Work Files with Extension NCD*.

Notes:

1. The `RECORD` option of the `READ WORK FILE` statement is not available for reading work files of format Entire Connection.
2. The operand format U (Unicode) is not supported for the work file types Entire Connection. If U is used with these work file types, a runtime error message is displayed.

Portable Format

Possible type: Portable.

The type Portable performs an automatic endian conversion of a work file when the work file is transferred to a different machine. For example, a work file written on a PC (little endian) can be read correctly on an RS6000 or HP machine (big endian). The endian conversion applies only to field formats I2, I4, F4, F8 and U. The floating point format is assumed to be IEEE. There are, however, slight differences in IEEE floating point representation by different hardware systems. As a rule, these differences apply only to infinity and NaN representations, which are normally not written into work files. Check the hardware descriptions if you are uncertain.

The files are always written in the machine-specific representation, so that a conversion is performed only if the file is read by a machine with different representation. This keeps performance as fast as possible.

There are no other conversions for this format apart from the conversions mentioned above.

When a `READ WORK FILE` statement is used for a dynamic variable, the variable is resized to the length of the current record.

Unformatted Format

Possible type: Unformatted.

The type Unformatted reads or writes a complete file with just one dynamic variable and just one record (for example, to store a video which was read from a database). No formatting information is inserted; everything is written and read just as it is.

CSV Format

Possible type: CSV (comma-separated values).

Note:
If you want to use the work file type CSV, you have to recatalog your sources using the `CATALOG` or `STOW` command. It is not possible to use the work file type CSV with generated programs of Natural Version 4.

The Natural fields are stored in a CSV work file as described below.

1. In the first step, the internal field data is converted into a readable format:

   - The field data of the internal Natural data formats B (binary), O (object handle), G (GUI handle) and C (attribute control) is copied to the record without field conversion. The data is taken as it is.

   - The field data of the internal Natural data format A (alphanumeric) is converted into the specified work file code page (see *Work Files* in the *Configuration Utility* documentation). If no work file code page is specified in the Configuration Utility, the default code page which is defined with the parameter `CP` is used and no conversion is done.

     The field data of the internal Natural data format U (Unicode), is converted into the specified work file code page (see *Work Files* in the *Configuration Utility* documentation) or, if no work file code page is specified, into the default code page which is defined with the parameter `CP`.

   - The values of the internal Natural formats D (date) and T (time) are converted into an alphanumeric output format. The `DTFORM` parameter is evaluated so that the user-specified date and time format is used.

   - The internal field values of the numeric types are converted into an alphanumeric output format.

2. In the second step, the field data in readable format is copied to the CSV work file record. The fields in the work file are separated by the specified separator character. If a field contains special characters, the field is delimited by double quotes. Each written record is terminated with a carriage return and line feed (CR/LF).

If you have defined that a header with the Natural field names is to be written to the work file (see *Work File Assignments* in the *Configuration Utility* documentation), the following applies:

- With the `WRITE WORK FILE` statement, a header line containing the field names of the first written record is stored in the first line of the work file. If subsequent CSV records contain a different number of fields, it may be possible that the header line does not correspond to these subsequent CSV records.

- With the `READ WORK FILE` statement, it is assumed that the first line of the CSV work file is the header line. Therefore, the first line is skipped (that is: the record data in the first line is not returned).

Special Considerations for Work Files with the Extension NCD

If files with the extension "NCD" are created by Entire Connection and are then read into Natural via the `READ WORK FILE` statement, it is required that the Entire Connection option **Keep trailing blanks** is activated in the session properties. See your Entire Connection documentation for further information.

Note:
When you create an NCD file using Entire Connection and load this file using the Object Handler, you may receive an error indicating that the source control record is missing. To avoid this, make sure that the option **Keep trailing blanks** is active when you create the NCD file.

The following considerations apply for work files in Entire Connection format:

- If an NCD file is read with a `READ WORK FILE` statement and the corresponding NCF format file is not available or contains invalid information, the NCD file is assumed to be an ASCII work file.
- When the `APPEND` attribute is used to append data to an NCD file, the record layouts (that is: the field format and length information which is written to the NCF format file) of the old and new data must match. If the record layouts are different, an error occurs during runtime.
- The maximum work-file record size for `WRITE WORK FILE VARIABLE` that can be handled by Entire Connection is 32767 bytes.
- If you have "old" work files with the extension "NCD", the extensions must be changed.
- Each of the following profile parameters must be set to the same value for both read and write operations:

  `DC` (decimal character)
  `IA` (input assign character)
  `ID` (input delimiter character)

- Remember that the range of possible values for floating point variables on a mainframe computer is different from that on other platforms. The possible value range for F4 and F8 variables on a mainframe is:

  $\pm 5.4 * 10^{-79}$ to $\pm 7.2 * 10^{75}$

  The possible value range on most other platforms for F4 variables is:

  $\pm 1.17 * 10^{-38}$ to $\pm 3.40 * 10^{38}$

  The possible value range on most other platforms for F8 variables is:

  $\pm 2.22 * 10^{-308}$ to $\pm 1.79 * 10^{308}$