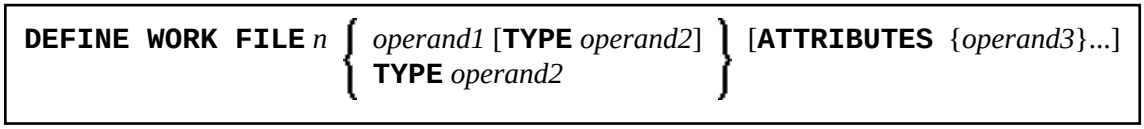
DEFINE WORK FILE



Note:
The elements shown in square brackets [...] are optional, however, at least one of them must be specified with this statement.

This chapter covers the following topics:

- Function
- Syntax Description
- Work File Name under z/OS Batch, TSO and Server
- Work File Name under z/VSE Batch
- Work File Name under VM/CMS
- Work File Name under BS2000/OSD Batch and TIAM
- Work File Name under CICS
- Work File Name under Com-plete/SMARTS

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: CLOSE WORK FILE | READ WORK FILE | WRITE WORK FILE

Belongs to Function Group: *Control of Work Files / PC Files*

Function

The statement `DEFINE WORK FILE` is used to assign a file name to a Natural work file number within a Natural application.

This allows you to make or change work file assignments dynamically within a Natural session or overwrite work file assignments made at another level.

When this statement is executed and the specified work file is already open, the statement will implicitly close the work file.

All work files to be used during a session must be preassigned to an access method by means of subparameter `AM` of profile parameter `WORK` or automatically by definition in the JCL.

Note:
For Unicode and code page support, see *Work Files and Print Files on Mainframe Platforms* in the *Unicode and Code Page Support* documentation.

Syntax Description

Operand Definition Table:

Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
operand1	C	S				A	U										yes	no
operand2	C	S				A	U										yes	no
operand3	C	S				A	U										yes	no

Syntax Element Description:

DEFINE WORK FILE *n*	*n* is the work file number (1 to 32). This is the number to be used in a `WRITE WORK FILE`, `READ WORK FILE` or `CLOSE WORK FILE` statement.

operand1	*operand1* is the name of the work file. As *operand1* you can specify the name of the dataset to be assigned to the work file number. *operand1* can be 1 to 253 characters long. You can specify either a logical or a physical dataset name. The possible format depends on the operating system environment and the access method defined by subparameter `AM` of profile parameter `WORK`. Some access methods do not support a work file name as *operand1*, e.g. `AM=COMP` and `AM=PC`. If *operand1* is not specified, the value of *operand1* is determined by taking the current name specified with the previously performed `DEFINE WORK FILE` statement for this work file number. If no previous `DEFINE WORK FILE` statement was performed, the name is taken from the Natural parameter module NATPARM. **Note:** If *operand1* is not specified, the behavior of Natural for Mainframes and Natural for Windows/UNIX/OpenVMS is different. Information on operating-system- or TP-monitor-dependent work file naming conventions is included in the following sections: ● Work File Name under z/OS Batch, TSO and Server ● Work File Name under z/VSE Batch ● Work File Name under VM/CMS ● Work File Name under BS2000/OSD Batch and TIAM ● Work File Name under CICS ● Work File Name under Com-plete/SMARTS	
TYPE *operand2*	*operand2* specifies the type of work file. The value of *operand2* is handled in a case insensitive way and must be enclosed in quotes or provided in an alphanumeric variable.	
	UNFORMATTED	A completely unformatted file. No formatting information is written (neither for fields nor for records). `UNFORMATTED` treats a work file as a byte-stream with no record boundaries. Note that type `UNFORMATTED` will be rejected by Entire Connection.
	FORMATTED	`FORMATTED` defines a regular record-oriented work file, which is subject to the same handling as in previous Natural versions.

ATTRIBUTES {*operand3*}...	**ATTRIBUTES Clause:** This clause makes sense only in Natural for Open Systems; in Natural for Mainframes it is ignored.

Examples:

```
DEFINE WORK FILE 17 #FILE TYPE 'UNFORMATTED'
#TYPE := 'FORMATTED'
DEFINE WORK FILE 18 #FILE TYPE #TYPE
```

Work File Name under z/OS Batch, TSO and Server

The following topics are covered:

- Work File Name - operand1
- Allocation and De-Allocation of Datasets
- Work Files in Server Environments

Work File Name - *operand1*

Under z/OS, for a work-file number that is defined with the access method `AM=STD`, *operand1* can be:

- a logical dataset name (DD name, 1 to 8 characters);
- a physical dataset name of a cataloged dataset (1 to 44 characters) or a physical dataset member name;
- a path name and member name of an HFS file (1 to 253 characters) in an MVS UNIX Services environment;
- a JES spool file class;
- `NULLFILE`.

Logical Dataset Names	For example:
	
	`DEFINE WORK FILE 21 'SYSOUT1'`
	
	The specified dataset `SYSOUT1` must have been allocated before the `DEFINE WORK FILE` statement is executed.
	
	The allocation can be done via JCL, CLIST (TCO) or dynamic allocation (SVC 99). For dynamic allocation you can use the application programming interface USR2021N, which is located in library SYSEXT.
	
	The dataset name specified in the `DEFINE WORK FILE` statement overrides the name specified with subparameter `DEST` of profile parameter `WORK`.
	
	Optionally, the dataset name may be prefixed by `DDN=` to indicate that it is a DD name. For example:
	
	`DEFINE WORK FILE 22 'DDN=MYWORK'`

Physical Dataset Names	For example:
	
	`DEFINE WORK FILE 23 'TEST.WORK.FILE'`
	
	The specified dataset must exist in cataloged form. When the `DEFINE WORK FILE` statement is executed, the dataset is allocated dynamically by SVC 99 with the current DD name and option `DISP=SHR`.
	
	If the dataset name is 8 characters or shorter and does not contain a period (.), it might be misinterpreted as a DD name. To avoid this, prefix the name with `DSN=`. For example:
	
	`DEFINE WORK FILE 22 'DSN=WORKXYZ'`
	
	If the dataset is a PDS member, you specify the PDS member name (1 to 8 characters) in parentheses after the dataset name (1 to 44 characters). For example:
	
	`DEFINE WORK FILE 4 'TEST.WORK.PDS(TEST1)'`
	
	If the specified member does not exist, a new member of that name will be created.

HFS Files	For example:
	
	`DEFINE WORK FILE 14 '/u/nat/rec/test.txt'`
	
	The specified path name must exist. When the `DEFINE WORK FILE` statement is executed, the HFS file is allocated dynamically. If the specified member does not exist, a new member of that name will be created.
	
	For the dynamic allocation of the dataset, the following z/OS path options are used:
	
	`PATHOPTS=(OCREAT,OTRUNC,ORDWR)`
	`PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)`
	`FILEDATA=TEXT`
	
	When an HFS file is closed, it is automatically de-allocated by z/OS (regardless of the setting of subparameter `FREE` of profile parameter `WORK`).
	
	To read an HFS file, you have to use the application programming interface `USR2021N` (dynamic dataset allocation) instead of the `DEFINE WORK FILE` statement, because of the `OTRUNC` option. This option resets the HFS file at the first read access, which results in an empty file.

JES Spool File Class	To create a JES spool dataset, you specify `SYSOUT=`*x* (where *x* is the desired spool file class). For the default spool file class, you specify `SYSOUT=*`.
	
	Examples:
	
	`DEFINE WORK FILE 10 'SYSOUT=A'`
	`DEFINE WORK FILE 12 'SYSOUT=*'`
	
	To specify additional parameters for the dynamic allocation, use the application programming interface `USR2021N` (dynamic dataset allocation) in the library `SYSEXT` instead of the `DEFINE WORK FILE` statement.

NULLFILE	To indicate a dummy dataset.

Allocation and De-Allocation of Datasets

When the `DEFINE WORK FILE` statement is executed and a physical dataset name, HFS file, spool file class or dummy dataset has been specified, the corresponding dataset is allocated automatically. If the logical file is already open, it will be closed automatically, except when the subparameter `CLOSE=FIN` of profile parameter `WORK` has been specified, in which case an error will be issued. Moreover, an existing dataset allocated with the same current DD name is automatically de-allocated before the new dataset is allocated. To avoid unneccessary overhead by unsuccessful premature opening of work files not yet allocated at the start of the program, work files should be defined with the subparameter `OPEN=ACC` (open at first access) of profile parameter `WORK`.

In the case of an HFS file, or a work file defined with the subparameter `FREE=ON` of profile parameter `WORK`, the work file is automatically de-allocated as soon as it has been closed.

As an alternative for the dynamic allocation and de-allocation of datasets, the application programming interface USR2021N (dynamic dataset allocation) in the library `SYSEXT` is provided. This also allows you to specify additional parameters for dynamic allocation.

Work Files in Server Environments

In server environments, errors may occur if multiple Natural sessions attempt to allocate or open a dataset with the same DD name. To avoid this, you either specify the work file with the subparameter `DEST=*` of profile parameter `WORK`, or you specify `DEFINE WORK FILE '*'` in your program before the actual `DEFINE WORK FILE` statement; Natural then generates a unique DD name at the physical dataset allocation when the first `DEFINE WORK FILE` statement for that work file is executed.

All work files whose DD names begin with `CM` are shared by all sessions in a server environment. A shared work file opened for output by the first session is physically closed when the server is terminated. A shared work file opened for input is physically closed when the last session closes it, that is, when it receives an end-of-file condition. When a work file is read concurrently, one file record is supplied to one `READ WORK FILE` statement only.

Work File Name under z/VSE Batch

Under z/VSE, for a work-file number that is defined with the access method `AM=STD`, *operand1* can be:

- a logical dataset name (DD name, 1 to 7 characters);
- `NULLFILE` (to indicate a dummy dataset).

Logical Dataset Names	For example:
	`DEFINE WORK FILE 21 'SYSOUT1'`
	The specified dataset `SYSOUT1` must have been defined in the JCL or in the z/VSE standard or partition labels.
	The dataset name specified in the `DEFINE WORK FILE` statement overrides the name specified with subparameter `DEST` of profile parameter `WORK`.
	Optionally, the dataset name may be prefixed by `DDN=` to indicate that it is a DD name. For example:
	`DEFINE WORK FILE 22 'DDN=MYWORK'`

NULLFILE	To allocate a dummy dataset, you specify `NULLFILE` as *operand1*: `DEFINE WORK FILE n 'NULLFILE'`

Work File Name under VM/CMS

Under VM/CMS, for a work file that is defined with the `AM=STD`, the same applies to *operand1* as under z/OS (see above) but with the following differences:

- Instead of dynamic allocation via MVS SVC 99, the CMS command `FILEDEF` is used to define a file.
- HFS files are not supported.

- JES spool classes are not supported.
- In addition, the following syntax is used:

  ```
  DEFINE WORK FILE n 'fname ftype fmode (options)'
  ```

  This generates the CMS command:

  ```
  FILEDEF ddname-n DISK fname ftype fmode (options)
  ```

- Moreover, the following syntax is allowed:

  ```
  DEFINE WORK FILE n 'FILEDEF=filedef-parameters'
  ```

  This generates the CMS command:

  ```
  FILEDEF ddname-n =filedef-parameters
  ```

  For example:

  ```
  DEFINE WORK FILE 5 'FILEDEF=TAP1 SL 2 VOLID BKUP08 (BLKSIZE 20000)'
  ```

  This generates the CMS command:

  ```
  FILEDEF CMWKF05 TAP1 SL 2 VOLID BKUP08
  ```

For a work file that is defined with `AM=CMS`, `DEFINE WORK FILE` can be used to change the destination. In addition to destinations that can be specified with subparameter `DEST` of the parameter macro `NTWORK`, a Rexx stem can be defined. For details, refer to *Print File and Work File Support* in the *Operations* documentation.

Work File Name under BS2000/OSD Batch and TIAM

Under BS2000/OSD, for a work-file number that is defined with the access method `AM=STD`, you can use *operand1* to specify a file name or a link name that is allocated to this work file.

In this case, *operand1* can have a length of 1 to 253 characters and one of the following meanings:

- a BS2000/OSD link name (1 to 8 characters)
- a BS2000/OSD file name (9 to 54 characters)
- a generic BS2000/OSD file name (wildcard)
- a BS2000/OSD file name and link name
- a generic BS2000/OSD file name and link name (wildcard)
- *DUMMY

The following rules apply.

1. File name and link name can be specified as positional parameters or keyword parameters. The corresponding keywords are `FILE=` and `LINK=`. Mixing positional and keyword parameters is allowed but not recommended.

2. A string with a length of 1 to 8 characters without commas is interpreted as a link name. This notation is compatible with earlier versions of Natural. Example:

   ```
   DEFINE WORK FILE 1 'W01'
   ```

   The corresponding definition with a keyword parameter is:

   ```
   DEFINE WORK FILE 1 'LINK=W01'
   ```

3. A string of 9 to 54 characters without commas is interpreted as a file name.

   Example:

   ```
   DEFINE WORK FILE 2 'NATURALvr.TEST.WORKFILE02'
   ```

   where *vr* stands for the Natural version and release number.

   The corresponding definition with a keyword parameter is:

   ```
   DEFINE WORK FILE 2 'FILE=NATURALvr.TEST.WORKFILE02'
   ```

4. The following input is interpreted without considering the length and therefore forms exceptions to Rules 2 and 3:

   - keyword input: `LINK=`, `FILE=`
   - `*DUMMY`
   - `NULLFILE` (equivalent to `*DUMMY`)
   - *
   - *,*

   Example: `DEFINE WORK FILE 7 'FILE=Y'` is a valid file allocation and not a link name, although the string of characters contains fewer than 9 characters.

5. Generic file names are formed as follows:

   ```
   Wnn.userid.tsn.date.time.number
   ```

   where

nn	is a work-file number
userid	is a Natural user-ID, 8 characters
tsn	is the BS2000/OSD TSN of the current task, 4 digits
date	is *DDMMYYYY*
time	is *HHIISS*
number	is a number, 5 digits

6. Generic link names are formed as follows:

   NWF*nnnnn*

   nnnnn is a 5-digit number that is increased by one after every generation of a dynamic link name.

7. Changing the file allocation for a work-file number causes an implicit CLOSE of the work file allocated so far.

You are strongly recommended, in all cases except when you only specify a link name (for example: W01), to work with keyword parameters. This avoids conflicts of interpretation with additional reports and is essential for file names with fewer than 9 characters.

Example:

```
DEFINE WORK FILE 3 'LINK=#W03'
DEFINE WORK FILE 3 'FILE=#W03'
```

Link Name	Example:
	`DEFINE WORK FILE 1 'LINKW01'`
	means the same as
	`DEFINE WORK FILE 1 'LINK=LINKW01'`
	A file with the LINK LINKW01 must exist at runtime. This can be created either using JCL before starting Natural or by dynamic allocation from the current application. For dynamic allocation, the application programming interface USR2029N (dynamic file allocation) in the library SYSEXT can be used. If, before execution, the link was active on another file, for example: W01, this will be released or retained depending on the value of the profile parameter FREE (possible values are ON and OFF). Release is done via an explicit RELEASE call to the BS2000/OSD command processor.

File Name	Example:
	`DEFINE WORK FILE 2 'NATURAL`*vr*`.TEST.WORK02'`
	means the same as
	`DEFINE WORK FILE 2 'FILE=NATURAL`*vr*`.TEST.WORK02'`
	where *vr* stands for the Natural version and release number.
	The file specified in *operand1* is set up using a FILE macro call and inherits the link name that was valid for the corresponding work file before execution of the DEFINE WORK FILE statement.

Generic File Name	Example:
	`DEFINE WORK FILE 21 '*'`
	means the same as
	`DEFINE WORK FILE 21 'FILE=*'`
	A file with a name created according to Rule 4 is set up using a `FILE` macro call and inherits the link name that was valid for the corresponding work file before execution of the `DEFINE WORK FILE` statement.
	`DEFINE WORK FILE 22 'FILE=*,LINK=WFLK22'`
	A file with a name created according to Rule 4 is set up with the specified link name, using a `FILE` macro call.

File Name and Link Name	Example:
	`DEFINE WORK FILE 11 'NATURAL`*vr*`.TEST.WORKF11,LNKW11'`
	means the same as
	`DEFINE WORK FILE 11 'FILE=NATURAL`*vr*`.TEST.WORKF11,LINK=LNKW11'`
	which means the same as
	`DEFINE WORK FILE 11 'FILE=NATURAL`*vr*`.TEST.WORKF11,LNKW11'`
	where *vr* stands for the Natural version and release number.
	The file given in *operand1* is set up with the specified link name, using a `FILE` macro call and allocated to the corresponding work-file number.

Generic File Name and Link Name	Example:
	`DEFINE WORK FILE 27 '*,*'`
	means the same as
	`DEFINE WORK FILE 27 'FILE=*,LINK=*'`
	A file with a file name and link name created according to Rule 4 and Rule 5 is set up using a `FILE` macro call and allocated to the specified work file 27.
	Note:
	When file name and link name are specified, the previous link name is not released, regardless of the value of the subparameter `FREE` of profile parameter `WORK`.

***DUMMY**	To indicate a dummy dataset.

Work File Name under CICS

For a work-file number defined with the access method AM=CICS, *operand1* can be a transient data or temporary storage queue name (1 to 8 characters), depending on subparameter TYPE of profile parameter WORK for the work file. For TYPE=TD, only the first 4 characters of *operand1* are honored and the transient data destination must be predefined to CICS.

For further information on work files, see also *Natural Print and Work Files under CICS* (in the *TP Monitor Interfaces* documentation).

Work File Name under Com-plete/SMARTS

Under Com-plete with the access method AM=SMARTS, PFS files are available. Any work file name can be assigned, even if it has not been defined to Natural. For example:

```
DEFINE WORK (14) '/nat/path/workfile'
DEFINE WORK (14) 'workfile'
```

It depends on the MOUNT_FS parameter of SMARTS whether the file is located on a SMARTS portable file system or on the native file system. The first element of the path (*/nat/*) determines the target file system.

If the string does not start with a slash (/), the path of the file is taken from the environment variable $NAT_WORK_ROOT.

The specified path name must exist. When the DEFINE WORK FILE statement is executed, the file is allocated dynamically. If the specified member does not exist, a new member of that name will be created.