System Files

Natural for OpenVMS stores objects in files accessible by operating system functions. Unlike Natural for Mainframes where the objects are stored in Adabas system files, Natural for OpenVMS stores the objects in specific directories on the disk. Thus, a database such as Adabas is not required to run Natural for OpenVMS.

This chapter covers the following topics:

System File Structure

By default, the Natural libraries are created as subdirectories below the Natural root directory of a specific Natural version. The subdirectories have the same names as the libraries.

The Natural objects are stored as files in the subdirectories. The file name for a Natural object has the following form:

file-name.NK*T*

file-name	This the name of the object. See also *Object Naming Conventions* in *Using Natural*.
N	The first character of the extension is always "N". It stands for "Natural".
K	The second character of the extension can be one of the following: S — for source files G — for generated programs R — for resources
T	The third character of the extension stands of the type of the object. For valid values, see the list below.

For example, the source program `TESTPROG` is stored as file *TESTPROG.NSP*, while the generated code for the map `TESTMAP` is stored as file *TESTMAP.NGM*.

Note:
The file name is not always identical to the object name. Both the current object name and the corresponding internal object name are documented in the file *FILEDIR.SAG*.

The following object types and the respective letters and numbers are used for the extensions available:

Letter or Number	Object Type
A	Parameter data area (PDA)
C	Copycode
D	DDM
G	Global data area (GDA)
H	Helproutine
L	Local data area (LDA)
M	Map
N	Subprogram
P	Program
S	Subroutine
T	Text
4	Class
5	Command processor
7	Function
8	Adapter

System Files FNAT and FUSER

The Natural system files `FNAT` (for system programs) and `FUSER` (for user-written programs) are located in different subdirectories.

`FNAT` assumes the following directory structure:

```
FNAT
 :.... LIBDIR.SAG
 :.... SYSTEM
          :.... FILEDIR.SAG
          :.... SRC
          :.... GP
          :.... ERR
          :.... RES
```

```
  └── SYS*
          └── FILEDIR.SAG
          └── SRC
          └── GP
          └── ERR
          └── RES
```

The file *LIBDIR.SAG*, which is only available for `FNAT`, contains information on all further installed Software AG products using Natural. This information can be displayed by using the system command `SYSPROD`.

`FUSER` assumes the following directory structure:

```
FUSER
  └── SYSTEM
          └── FILEDIR.SAG
          └── SRC
          └── GP
          └── ERR
          └── RES
  └── user-library1
          └── FILEDIR.SAG
          └── SRC
          └── GP
          └── ERR
          └── RES
```

The name of a user library must not start with "SYS".

The directory structure is generated during the installation of Natural. The directories representing the system and user libraries contain the following:

- ***FILEDIR.SAG***
  This file contains internal library information used by Natural. For further information, see *The File FILEDIR.SAG* below.
- ***SRC***
  This subdirectory contains the Natural source objects stored in the library.
- ***GP***
  This subdirectory contains the generated Natural programs stored in the library.
- ***ERR***
  This subdirectory contains the error messages stored in the library.
- ***RES***
  This subdirectory contains the private and shared resources stored in the library.

DDMs can be stored in local libraries. If DDMs are used by a program, Natural first searches the current library, then the steplibs, and then the library `SYSTEM`. If the DDMs are not found, the program does not compile and displays an error message. However, if FDDM mode has been activated, Natural searches for the DDMs only in the system file `FDDM`.

The paths to the system files `FNAT`, `FUSER` and `FDDM` are defined in the Configuration Utility. System files are version-dependent. Therefore, Natural can only access system files of the current Natural version. It is recommended that you only have one `FNAT` system file. It is possible, however, to define several `FUSER` system files (for example, when you have different development areas for different purposes).

System File FDDM

The system file `FDDM` is a container in which all DDMs can be stored.

`FDDM` assumes the following directory structure:

```
FDDM
 └── SYSTEM
      └── FILEDIR.SAG
      └── SRC
      └── GP
```

By default, the system file `FDDM` is not active. If you want to use it, you have to activate FDDM mode as described below.

- Activating FDDM Mode
- Migrating DDMs to the System File FDDM
- Checking whether the System File FDDM is Used

Activating FDDM Mode

If FDDM mode is activated (both database ID and file number do not equal 0 in the global configuration file), all DDMs are stored and read in the system file `FDDM`. DDMs stored in libraries will no longer be accessible from Natural. This is similar to the mainframe, where all DDMs are stored in the system file `FDIC`.

If the `FDDM` system file is undefined in the global configuration file, the DDMs are stored in the Natural libraries `FUSER` and `FNAT`.

To activate FDDM mode

1. Create an empty directory in which the DDMs are to be stored in FDDM mode. The directory can have any name which corresponds to the Natural naming conventions.

2. Invoke the Configuration Utility.

3. In the global configuration file (category **System Files**), assign a database ID and file number for the system file `FDDM` and define the path to the directory that you have created in the first step.

4. Open the required parameter file.
5. Locate the parameter `FDDM`.

   Tip:
   Locate this parameter by searching for "FDDM". See *Finding a Parameter* in the *Configuration Utility* documentation for further information.
6. For the parameter `FDDM`, specify the same database ID and file number that you have defined in the global configuration file.
7. Save your changes.
8. Migrate all required DDMs to the system file `FDDM` as described below.

Migrating DDMs to the System File FDDM

All DDMs that are to be available in FDDM mode must be contained in the system file `FDDM`. Especially the example DDMs delivered with Natural in library `SYSEXDDM` must be available in the system file `FDDM`.

For migration of DDMs to the `FDDM` system file, you can choose between different alternatives:

- You can use the Object Handler which supports the `FDDM` system file and offers the possibility to migrate the DDMs into the `FDDM` system file. The DDMs can be unloaded from the Natural libraries and can be stored into the `FDDM` system file in the active Natural session.

  Important:
  To migrate a complete development environment, it is recommended to use the Object Handler.
- It is also possible to migrate the DDMs with the copy or move function of the `SYSMAIN` utility. In this case, it is required that the `FDDM` parameter is first deactivated so that your old environment is used again.

These alternatives are described below in detail.

Note:
The `INPL` utility loads DDMs either to Natural libraries if FDDM mode is not active or to the system file `FDDM` if FDDM mode is active. This may have some impact if the loaded INPL files are intended to work in both modes. It may be necessary that the DDMs are available in the Natural libraries as well as in the `FDDM` system file.

To migrate DDMs to the system file `FDDM` using the Object Handler

1. Activate FDDM mode as described above.
2. Start Natural using the modified parameter file (that is, the parameter file in which path for the parameter `FDDM` has been defined).
3. Issue the direct command `SYSOBJH` to invoke the Object Handler.

The following steps assume that you use the Object Handler wizards.

4. In the main menu, mark the **Unload** function and press ENTER.
5. In the resulting screen, mark the option **Unload objects into Natural work file(s)** and press ENTER.
6. In the resulting screen, mark the option **Set additional options** and press ENTER.
7. In the resulting screen, deactivate the option **Use FDDM file for processing DDMs** and press ENTER to return to the previous screen.

   This activates your old environment (which contains the DDM to migrated). If you do not deactivate this option, you cannot access the DDMs that are to be migrated.
8. Press ENTER repeatedly until the screen is shown in which the object type for the unload has to be selected.

   The option **Natural library objects only** is selected by default. This option is required for the next steps.
9. Press ENTER.
10. In the resulting screen, enter an asterisk (*) in the fields **Library** and **Object name**. In addition, mark the field **More detailed specification of objects**. Press ENTER.
11. In the resulting screen, deactivate the options **Error messages** and **Shared resources**. In the **Natural types** field, enter "V" and press ENTER.
12. Press ENTER to display the command that is to be processed.
13. Press ENTER to start the unload function.
14. When the objects have been unloaded, return to the main menu.
15. In the main menu, mark the **Load** function and press ENTER.
16. In the resulting screen, mark the option **Load objects from Natural work file(s)** and press ENTER.
17. In the resulting screen, mark the option **Set additional options** and press ENTER.
18. In the resulting screen, activate the option **Use FDDM file for processing DDMs**.

    This activates your new environment containing the `FDDM` system file.

    Note:
    In different libraries, DDMs can exist with identical names. To prevent overwriting DDMs in the `FDDM` system file and to detect DDMs with identical names, it is recommended to load the DDMs with the **Do not replace** option. This option is located on the same page as the option **Use FDDM file for processing DDMs**.
19. Press ENTER to return to the previous screen.

20. Press ENTER repeatedly until the screen is shown in which the object type for the load has to be selected.

    The option **Load all option from the work file** is selected by default. This option is required for the next steps.

21. Press ENTER.

    The command that is to be processed is now shown.

22. Press ENTER to load the objects.

Checking whether the System File FDDM is Used

When you have migrated all DDMs to the system file `FDDM`, you can check whether `FDDM` is used.

To check whether `FDDM` is used

1. Start Natural.
2. Issue the system command `SYSPROF`.
3. If the `FDDM` file is displayed, Natural will access only DDMs stored in this system file.

   If the `FDDM` file is not displayed or if the expected files are not displayed, revise the parameter file used for your session.

Important Information and Warnings

A Natural developer must have read, write and delete rights for all objects.

An end-user must only have read rights for the generated programs (and in some special cases also read rights for the sources).

Do not access Natural files with operating system utilities. These utilities might modify and destroy the Natural directory information.

The use of an external editor is not recommended as code page conflicts may arise. These conflicts can - but not necessarily must - deteriorate your source code.

Do not store private data files in the directories `FNAT`, `FUSER` and `FDDM`, since Natural may delete or modify them in an unexpected way.

Do not use one of the directories `FNAT`, `FUSER` and `FDDM` as working directories for your OpenVMS applications, since this can cause problems when issuing Natural system commands.

The file name (i.e path including file name in 8.3 format) of any object accessed by Natural must not exceed 255 bytes.

The File *FILEDIR.SAG*

The file *FILEDIR.SAG* supports up to 60000 objects. It contains internal library information used by Natural including the programming mode of an object (structured or reporting) and internally converted object names. These internal object names are automatically created when storing Natural objects to disk with:

- names longer than 8 characters (which can be the case with DDMs);
- names containing any special character supported by Natural but not by the operating system.

Internal object names are unique and consist of an abbreviation of the current object name and an arbitrary number. Both the current object name and the corresponding internal object name are documented in *FILEDIR.SAG*.

Even if an object is located in the correct directory, it can only be used by Natural after this library information is included in *FILEDIR.SAG*. For objects created within Natural, the library information is included automatically. For all other objects, the **Import** function of the `SYSMAIN` utility should be used.

The utility `FTOUCH` can be used to update *FILEDIR.SAG* without entering Natural.

Portable Natural System Files

Starting with Natural Version 6.2, the directory file *FILEDIR.SAG* in a Natural library as well as the Natural error message files are created in a portable platform-independent format. This offers, for example, the possibility of exchanging `FUSER` libraries between different Windows, UNIX and OpenVMS platforms simply by copying the libraries via operating system commands.

The `FNAT` system file belongs to a Natural installation and is both version-specific and platform-specific. Therefore, it is not recommended to share `FNAT` system files among different platforms. Especially the `FNAT` system file on a Windows platform contains a completely different set of utilities as the `FNAT` system file on some UNIX or OpenVMS platforms.

Although it is now possible to share an `FUSER` system file among different platforms, this possibility should by handled with care because Natural’s locking mechanism does not cross machine boundaries and hence it would be possible for two Natural sessions on different platforms to modify the same object at the same time with unpredictable results.

All libraries that are newly created as of Natural Version 6.2 have a new *FILEDIR.SAG* structure. Especially the `FNAT` system file delivered and installed as of Natural Version 6.2 has only libraries with the new structure.

The following topics are covered below:

- Language-dependent Objects
- Migrating an Old FILEDIR.SAG File

Language-dependent Objects

When the application to be ported uses the system variable `*LANGUAGE`, you have to take notice of the following information.

Almost all Natural objects are stored in the system file with a name which contains only upper-case characters. An exception are the language-dependent objects (that is: the objects which have been created for a specific language). Language-dependent objects may contain lower-case characters in their names. Since Windows is a case-preserving operating system (whereas UNIX is a case-sensitive operating system), it may happen that names which have been created under UNIX cause a conflict in Windows, or that an application which has been developed under UNIX yields unexpected results in Windows.

Note:
OpenVMS behaves similar to Windows. It does not distinguish between upper-case and lower-case characters. However, file names are always created with upper-case characters.

Example

The command `SAVE PGM&` creates an object where the object name contains the language identifier. The resulting object name depends on the setting of `*LANGUAGE`:

Setting of `*LANGUAGE`	An object with the following name is created
33	`PGMX` (with an upper-case X)
59	`PGMx` (with a lower-case x)

The separate objects which have been created under UNIX (*PGMX.NGP* and *PGMx.NGP*) get entries in the file *FILEDIR.SAG* with the names `PGMX` and `PGMx`. These two objects will be treated differently, depending on the environment in which Natural is being executed:

- When you execute `PGMX` with Natural for UNIX, the file *PGMX.NGP* is loaded into the buffer pool and executed.
- When you execute `PGMX` with Natural for Windows, either the file *PGMX.NGP* or *PGMx.NGP* is loaded into the buffer pool and executed. This is because Windows does not distinguish between these two objects and treats them as one and the same object. Thus it may be possible that applications which share an `FUSER`, or a copy of such an `FUSER`, behave in a different manner.

Migrating an Old *FILEDIR.SAG* File

Starting with version 6.2, Natural can read old platform-specific *FILEDIR.SAG* files on the platform for which they were formerly generated, but it cannot modify old *FILEDIR.SAG* files. When a library with an old *FILEDIR.SAG* file is accessed for modification, *FILEDIR.SAG* is converted into the new format before any further modification takes place.

Important:
It is recommended that you create a backup copy of the old `FUSER` system file before executing any of the steps (which lead to a conversion of *FILEDIR.SAG*) listed below.

There are a number of possibilities that lead to a conversion of an old *FILEDIR.SAG* file into the new format:

- As of version 6.2, Natural automatically converts an old *FILEDIR.SAG* format when a modify access is made. This is completely transparent for the user; it has not to be forced in any way. A modify access is, for example, a `SAVE` of a new source, a `CATALOG` of a source or a `CATALL`. A copy operation modifies the destination library and hence the *FILEDIR.SAG* file of the destination library. A move operation additionally modifies the source library because the object has to be deleted there. In any case, the original *FILEDIR.SAG* file is saved as *FILEDIR.BCK* in the library directory.

- All libraries that are to be converted can be unloaded with the Object Handler (`SYSOBJH`). When the resulting work file is reloaded into a new `FUSER` system file, all libraries are generated with the new *FILEDIR.SAG* structure.

- The utility `FTOUCH` provides the option `convert` which converts the *FILEDIR.SAG* file of the specified library into the new structure. The original *FILEDIR.SAG* file remains in the library directory as *FILEDIR.BCK*. Refer to the description of the `FTOUCH` utility for the syntax and a usage example.

- The copy function of the utility `SYSMAIN` can be used to copy a complete set of libraries from an old format `FUSER` system file into a new `FUSER` system file. In the destination `FUSER`, the *FILEDIR.SAG* files are automatically generated with the new structure.

The Natural versions prior to Natural Version 6.2 cannot access libraries with a new portable *FILEDIR.SAG* file. Therefore, system files cannot be shared between Natural Version 6.2 or above and an older version of Natural. This is only possible when all libraries are still in the old *FILEDIR.SAG* format and when no modify access has ever been made. In a production environment, it is possible, for example, to make use of an `FUSER` from Natural Version 6.1 with Natural Version 6.2 when no modification on the libraries is made. However, it is recommended not share system files between Natural Version 6.2 or above and an older version

Natural Root Directory

During the installation, the logical names `NATDIR` and `NATVERS` are created automatically. They point to Natural’s version-dependent root directory.

```
NATDIR:['F$Trnlnm("NATVERS")']
```

Using NFS to Store Natural Libraries

When you use NFS (Network File System) to store Natural libraries, you can run into problems when the directories in which the Natural libraries are stored are mounted via NFS from a file server in your network.

The reason for this is the need to lock the *FILEDIR.SAG* file stored in each library during update operations of Natural objects.

If your NFS locking is incompatible or not properly set up between the involved platforms, Natural can hang in an uninterruptible state while waiting for NFS locking requests to be processed. These requests are generally logged on the consoles of the involved systems or in some other system-dependent log file.

The work-around to solve this problem is to store Natural libraries only on local disks if problems with a hanging and uninterruptible nucleus occur.