Logical File Names

Copyright

Icons in Body Text

Icon	Meaning
⚠	Caution
✿	Example
💡	Note
⬆	Recommendation
◇	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Logical File Names

Many applications use files for storage or communication, and access them using the ABAP commands **`open dataset`**, **`transfer dataset`**, and so on. When these operations are performed, the ABAP commands require physical file names in the syntax of the relevant operating system. You can link physical file names with logical file names. In this way, you can write platform-independent applications, and predefine specific file names or directories for users.

Overview

The following are examples of scenarios that use logical file names:

- The application defines the logical file name in the coding or customizing. The application uses FILE_GET_NAME [Page 10] to determine a physical file name from the logical file name.
- The application allows the specification of a logical file name using the user interface. You can configure the logical file names permitted for this context using aliases (see Defining Aliases [Page 18]). The specified logical file name is translated to a physical file name using FILE_GET_NAME.
- The application allows the specification of a physical file name using the user interface. The application uses FILE_VALIDATE_NAME [Page 13] to check this physical file name against a logical file name defined in the application. Aliases are also supported in this context.

You define the logical file names and paths in Customizing. To do this, you can use the Implementation Guide section *SAP NetWeaver* → *Application Server* → *System Administration* → *Platform-Independent File Names*, or the transactions **FILE** and **SF01**.

The definitions used by SAP applications are delivered with the system and are adapted when they are imported. For information on which application uses which logical file names, see the application-specific documentation. You can add further definitions.

Delivery of Predefined Logical File Names

SAP delivers a predefined logical file name for every access with which users can specify a physical or logical file name on a selection screen.

The system can only check the physical or logical file name specified by the user if one of the following conditions is fulfilled:

- You are assigning a physical path and/or file name to the delivered logical file name.
- You are assigning an alias to the delivered logical file name, for which a physical path and/or file name was defined.

If one of these conditions is fulfilled then validation is active for the relevant predefined logical file name.

SAP usually only delivers the definition of logical file names; the physical paths must be maintained by the system administrator. The system can only check file names and therefore ensure security if physical paths are specified.

If the validation for a delivered logical file name is not active, the corresponding application creates an entry in the Security Audit Log when it is executed. You can obtain information from the Security Audit Log about which accesses to the file system of the application server have not yet been secured.

Defining Logical File Names

The conversion of a logical file name into a platform-specific file name is controlled by multilevel definitions that are stored in tables.

Definition of the Logical File Name

- **Logical File Name**

  Platform-independent descriptive name of a file. The system creates this name on a cross-client basis, but you can also define client-specific logical file names. Logical file names allow, on the one hand, platform-independent programming, and on the other hand, validation of file names.

- **Physical File Name**

  Platform-dependent name of a file. Precisely one physical file name is stored for each operating system syntax group for each logical file name.

- **Data Format**

  Describes the format of the data.

  In addition to the usual data formats such as ASCII or binary data format, you can also select the format **DIR**. By selecting the format **DIR**, you specify that the associated logical file name does not refer to a standard file, but rather to a **check directory for validating file names**. For more information about working with the **DIR** format, see the section about the import parameter INCLUDING_DIR in Function Module FILE_VALIDATE_NAME [Page 13].

- **Work Area**

  Describes the content assignment of the affected file.

- **Logical Path**

  A logical path is a descriptive platform-independent name for a path. A logical file name usually has a logical path; only then can different physical paths be created for the logical file names, depending on the operating system platform.

Definition of the Logical Path

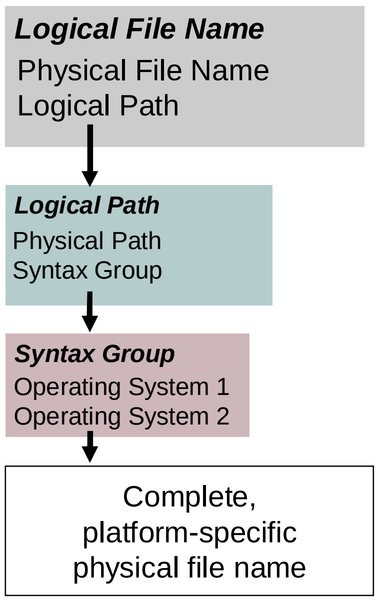
- **Physical Path**

  Platform-dependent name of a logical path for a specific syntax group. Platform-specific physical paths are specified for one or more syntax groups for a logical path.

- **Operating Systems and Syntax Groups**

  All operating systems used in a system configuration are assigned to syntax groups. A syntax group is the collective term for a set of operating systems with the same syntax for file names and paths. The definition of a syntax group specifies, for instance, how long file names may be, and whether file name extensions are permitted or not.

The graphic below shows the relationships between these objects, which convert a logical file name into a physical file name:



Parameters in Physical File Names and Paths

Physical file names and paths can contain the following reserved words, enclosed in angle brackets, as placeholders. The system replaces the placeholders with current values at runtime.

Reserved Words

Reserved Word	**Replacement Value**
<OPSYS>	Operating system in accordance with function module parameter (see below)
<INSTANCE>	Application instance
<SYSID>	Application name in accordance with system field SY-SYSID.
<DBSYS>	Database system in accordance with system field SY-DBSYS
<SAPRL>	Release in accordance with system field SY-SAPRL
<HOST>	Host name in accordance with system field SY-HOST
<CLIENT>	Client in accordance with system field SY-MANDT
<LANGUAGE>	Logon language in accordance with system field SY-LANGU
<DATE>	Date in accordance with system field SY-DATUM
<YEAR>	Year in accordance with system field SY-DATUM, four characters
<SYEAR>	Year in accordance with system field SY-DATUM, two characters
<MONTH>	Month in accordance with system field SY-DATUM
<DAY>	Day in accordance with system field SY-DATUM
<WEEKDAY>	Weekday in accordance with system field SY-FDAYW
<TIME>	Time in accordance with system field SY-UZEIT
<STIME>	Hour and minute in accordance with system field SY-UZEIT
<HOUR>	Hour in accordance with system field SY-UZEIT
<MINUTE>	Minutes in accordance with system field SY-UZEIT
<SECOND>	Seconds in accordance with system field SY-UZEIT
<PARAM_1>	External parameter 1 passed in function call
<PARAM_2>	External parameter 2 passed in function call
<PARAM_3>	External parameter 3 passed in function call
<P=name>	Value of a profile parameter in the current system
<V=name>	Value of a variable in the variable table
<F=name>	Return value of a function module

All physical paths must contain the reserved word <FILENAME> as a placeholder for the physical file name.

If you include parameters of this type in physical file or path names, you are supporting differentiated but standardized file names. For example, the parameter <TIME> can be useful if you save a logical file more than once in a short time. IN addition to the system field values, you can assign names flexibly, especially if you use the last parameters listed:

- You can use <PARAM_1> to <PARAM_3> to put values in file or path names that are passed explicitly when the application program calls the function module FILE_GET_NAME.
- You can use <P=name> to include the values of profile parameters in the current system. If you execute the report RSPARAM, you receive a list of profile parameters and their values.

- You can use <V=name> to include the values of variables that you defined in the control tables with transaction FILE.
- You can use <F=name> to include return values from function modules. The name of a function module used here must have the prefix “FILENAME_EXIT_”. Note that the function module in the reserved word is only addressed with the part of its name after this prefix. For example, if you use the function module FILENAME_EXIT_EXAMPLE, the placeholder is <F=EXAMPLE>. The function module must have the export parameter OUTPUT, and you must not specify a reference type for it. Import parameters must have default values. Table parameters are not supported.

Function Module FILE_GET_NAME

Application programs use logical file names in connection with the function module FILE_GET_NAME. Based on the definitions stored in Customizing, the function module creates the relevant physical name for a logical file name at runtime.

The following table gives an overview of the import and export parameters as well as the exception conditions of the function module.

Interfaces of the Function Module FILE_GET_NAME

Import Parameter	**Function**
CLIENT	Logical file names can also be defined as client-specific. The function module uses the definition of the client specified by this parameter. The default value is the current client as specified in system field SY- MANDT.
LOGICAL_FILENAME	Passes the logical file name. (Uppercase)
OPERATING_SYSTEM	Specifies the operating system for which the file name is to be determined. The default value is the operating system of the application server, as specified in system field SY OPSYS.
PARAMETER_1 PARAMETER_2 PARAMETER_3	Can pass any values to the placeholders labeled <PARAM_1> to <PARAM_3> in the physical file and path names.
USE_PRESENTATION_SERVER	Specifies that the file name is to be determined based on the operating system of the presentation server. The operating system imported using the parameter OPERATING_SYSTEM is not to be used.
WITH_FILE_EXTENSION	Specifies that the file format defined for the logical file name is to be appended to the physical file name as a file name extension. Exception: If the data format of the logical file name is DIR, the system ignores the parameter WITH_FILE_EXTENSION (see parameter INCLUDING_DIR).
USE_BUFFER	Specifies that the control tables are to be buffered.
ELEMINATE_BLANKS	Specifies whether all spaces are to be removed from the returned file name.
INCLUDING_DIR	In most application cases, the application that calls the function module FILE_GET_NAME requires a complete physical file name for additional processing, for example, for

	use with OPEN DATASET or similar commands. The data format DIR was introduced as part of the validation of user entries. If a logical file name has the data format DIR, the result of FILE_GET_NAME is a physical path name. By default, the indicator INCLUDING_DIR is not set. In this way, the calling program specifies that it requires a physical file name. Since a logical file name with the data format DIR does not fulfill this requirement, the system triggers the exception FILE_NOT_FOUND. If the calling application sets the INCLUDING_DIR indicator, it specifies that it can also process a physical file path as the result of FILE_GET_NAME. In this case the determined path name is returned to the calling application. If the INCLUDING_DIR indicator is set, the system ignores the parameter WITH_FILE_EXTENSION. It may return an empty file name to the calling application. In this case, the logic does not execute EMERGENCY_FLAG, in accordance with the description of the export parameter. For more information about validating user input, see the **Checking Directories** section of Function Module FILE_VALIDATE_NAME [Page 13].

Export Parameter	**Function**
EMERGENCY_FLAG	If the value is not SPACE, the system could not determine a physical file name for the logical file name for this operating system from the specifications in Customizing (for more information, see the note in connection with the interface documentation). In this case, the system uses the path specified in the profile parameter DIR_GLOBAL as the physical path.
FILE_FORMAT	This parameter contains the file format defined for the logical file name. You can use this parameter, for example, to specify the mode in which the file is to be opened.
FILE_NAME	This parameter contains the complete physical file name, including the path.

Exception	**Function**
FILE_NOT_FOUND	Triggered if the logical file name is not defined.
OTHERS	Triggered if other errors occur.

The following are possible reasons for the function module being unable to determine a physical path for this operating system (see parameter EMERGENCY_FLAG):

- The operating system is not defined in the control tables.
- The operating system is defined but not assigned to a syntax group.
- No physical path is assigned to the logical path for syntax group.
- No logical path is assigned to the logical file name.

Example

The following definitions exist for the logical file DATA_FILE and the logical path DATA_PATH:

DATA_FILE	Physical File	File<PARAM_1>	
	Data Format	BIN	
	Logical Path:	DATA_PATH	
DATA_PATH	Syntax Group:	UNIX	Physical Path: /tmp/<FILENAME>
	Syntax Group:	Microsoft Windows NT	Physical Path: c:\tmp\<FILENAME>

The operating system of the application server is assigned to syntax group UNIX, the operating system of the presentation server is assigned to the syntax group Windows NT. The two following calls of the function module would produce the specified return values.

Example 1

```
a.  CALL FUNCTION 'FILE_GET_NAME'
b.       EXPORTING
               LOGICAL_FILENAME        = 'DATA_FILE'
c.             PARAMETER_1             = '01'
d.       IMPORTING
               EMERGENCY_FLAG          = LD_EMERGENCY_FLAG
               FILE_FORMAT             = LD_FILE_FORMAT
               FILE_NAME               = LD_FILE_NAME
e.       EXCEPTIONS
               FILE_NOT_FOUND          = 1
               OTHERS                  = 2.
```

Return Values:

Variable	**Contents**
ld_emergency_flag	Space
ld_file_format	BIN
ld_file_name	/tmp/file01

Example 2

```
f.  CALL FUNCTION 'FILE_GET_NAME'
g.       EXPORTING
              LOGICAL_FILENAME         = 'DATA_FILE'
            USE_PRESENTATION_SERVER    = 'X'
            WITH_FILE_EXTENSION        = 'X'
h.       IMPORTING
            EMERGENCY_FLAG             = LD_EMERGENCY_FLAG
            FILE_FORMAT                = LD_FILE_FORMAT
            FILE_NAME                  = LD_FILE_NAME
i.       EXCEPTIONS
            FILE_NOT_FOUND             = 1
            OTHERS                     = 2.
```

Return Values:

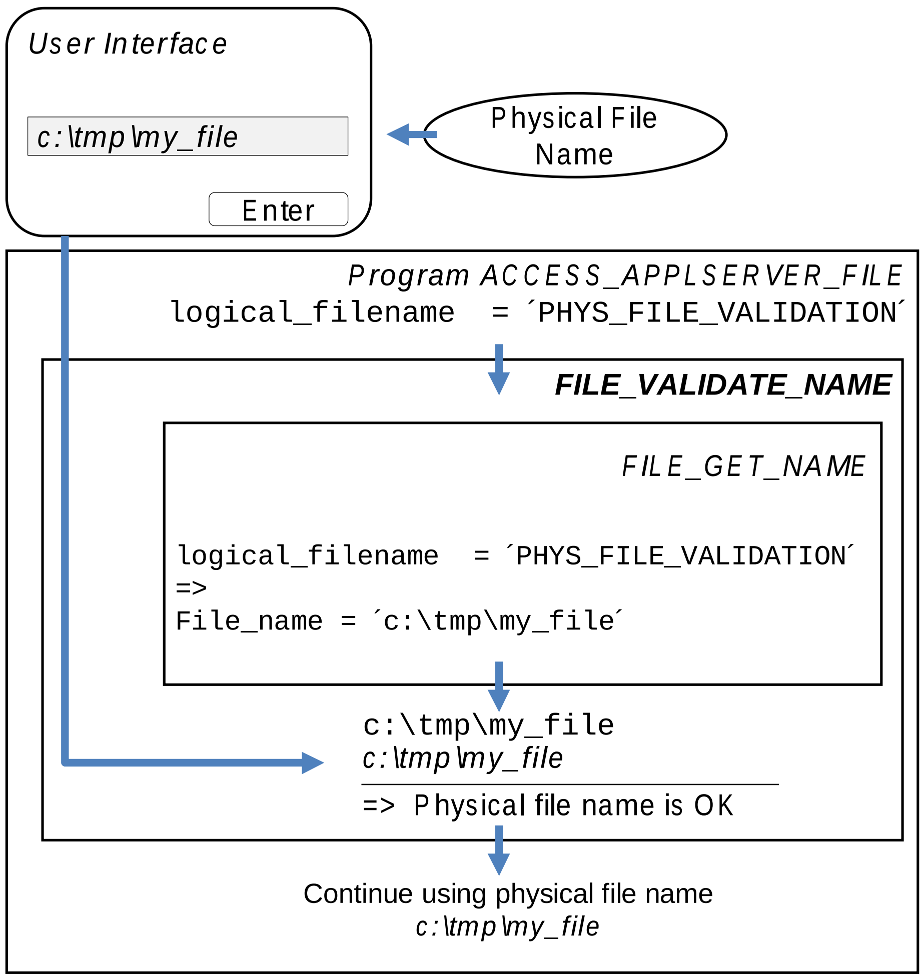
Variable	Contents
ld_emergency_flag	Space
ld_file_format	BIN
ld_file_name	C:\tmp\FILE.BIN

Function Module FILE_VALIDATE_NAME

Using a logical file name, the function module FILE_VALIDATE_NAME checks if a physical file name corresponds to the rules of the logical file name.

Example

The figure below uses a simplified example to illustrate the functional principle of FILE_VALIDATE_NAME. In this example, the operating system of the application server is *Microsoft Windows*.


User Interface
c:\tmp\my_file
Enter
Physical File Name
Program ACCESS_APPLSERVER_FILE
logical_filename = ´PHYS_FILE_VALIDATION´
FILE_VALIDATE_NAME
FILE_GET_NAME
logical_filename = ´PHYS_FILE_VALIDATION´
=>
File_name = ´c:\tmp\my_file´
c:\tmp\my_file
c:\tmp\my_file
=> Physical file name is OK
Continue using physical file name
c:\tmp\my_file

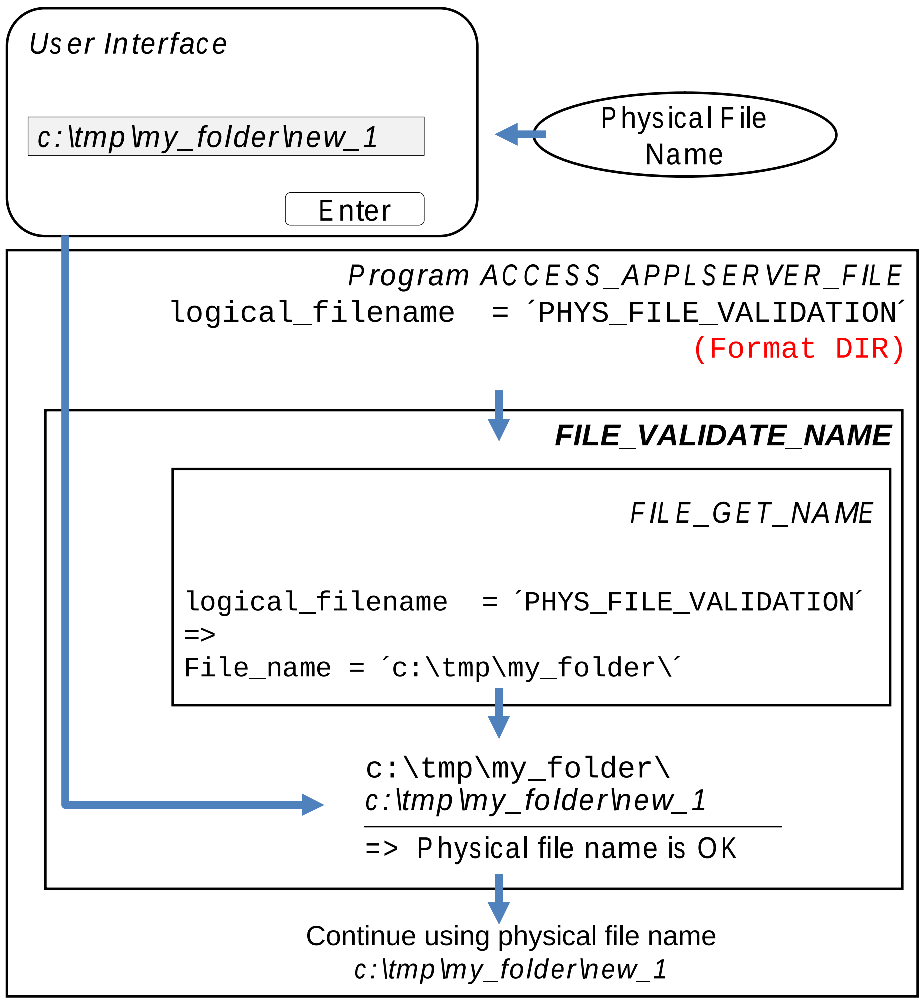
The example program ACCESS_APPLSERVER_FILE knows the logical file name *PHYS_FILE_VALIDATION* as a constant value.

If a user accesses the file *c:\tmp\my_file* with the program, the following steps are processed:

- The system calls the function module FILE_VALIDATE_NAME.
- This calls the function module FILE_GET_NAME. The system uses this function module to determine the complete physical file name for the logical file name and the appropriate operating system. The function module returns this value to FILE_VALIDATE_NAME.
- The system compares the two physical file names – on the one hand, from the input on the selection screen, and on the other hand, the value returned from calling FILE_GET_NAME.
- If the two file names match, the user can access the file.
- If the two file names do not match, the function module FILE_VALIDATE_NAME triggers the VALIDATION_FAILED exception. The program can deny the user access to the file.

Checking Directories

You do not need to restrict the check to specific file names, but can allow whole areas of a file system to be used as the access target. In this case, the system performs the check only for the path of the stored directory:

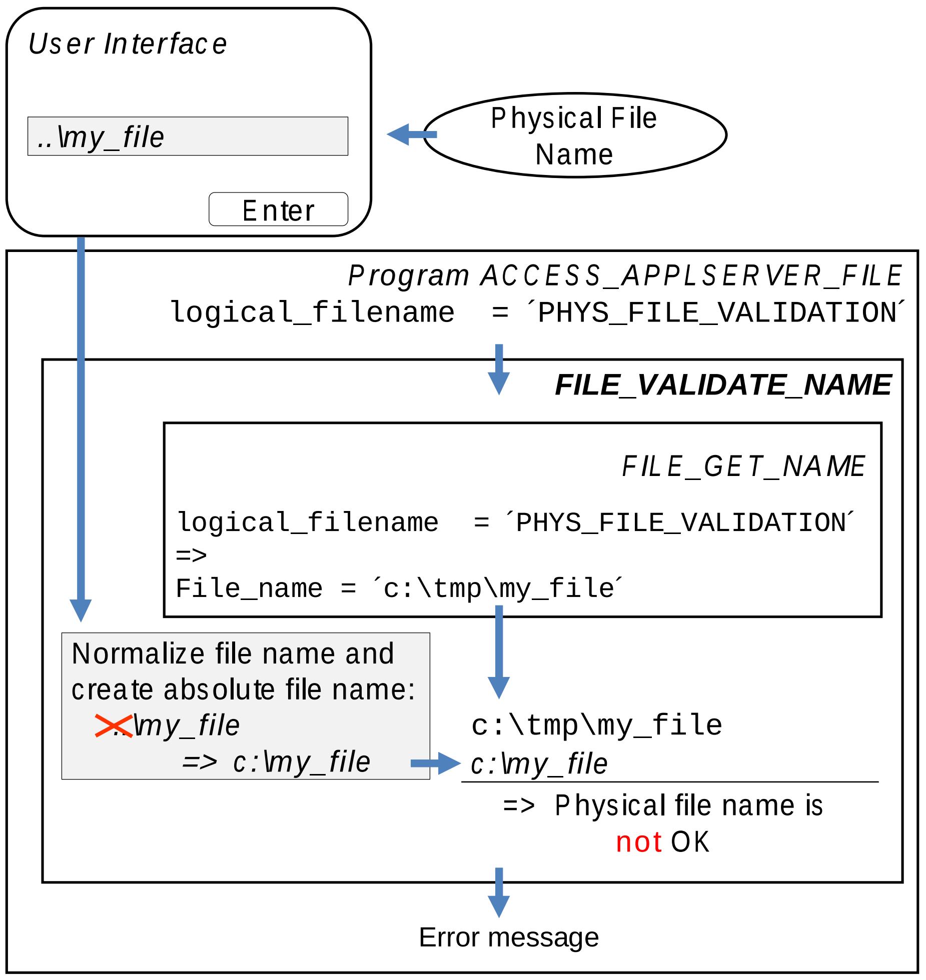

User Interface
c:\tmp\my_folder\new_1
Enter
Physical File Name
Program ACCESS_APPLSERVER_FILE
logical_filename = ´PHYS_FILE_VALIDATION´
(Format DIR)
FILE_VALIDATE_NAME
FILE_GET_NAME
logical_filename = ´PHYS_FILE_VALIDATION´
=>
File_name = ´c:\tmp\my_folder\´
c:\tmp\my_folder\
c:\tmp\my_folder\new_1
=> Physical file name is OK
Continue using physical file name
c:\tmp\my_folder\new_1

The prerequisite for this is that the logical file name has the data format **`DIR`**.

Normalization and Absolute File Names

Function module FILE_VALIDATE_NAME also performs another function:

If a user has specified a physical file name that contains operating system commands for navigating in the file system, the system interprets this navigation before the comparison with the result from FILE_GET_NAME. The system also converts the physical file name that was found to an absolute file name to allow a proper comparison. In the following example the global work directory is defined as *c:\tmp*.


User Interface
..\my_file
Enter
Physical File Name
Program ACCESS_APPLSERVER_FILE
logical_filename = ´PHYS_FILE_VALIDATION´
FILE_VALIDATE_NAME
FILE_GET_NAME
logical_filename = ´PHYS_FILE_VALIDATION´ =>
File_name = ´c:\tmp\my_file´
Normalize file name and create absolute file name:
..\my_file
=> c:\my_file
c:\tmp\my_file
c:\my_file
=> Physical file name is not OK
Error message

This ensures that the user can access only the intended files in this context.

Interfaces of Function Module FILE_VALIDATE_NAME

Import Parameter	**Function**
CLIENT	Logical file names can also be defined as client-specific. The system uses the definition of the client specified by this parameter. The default value is the current client as specified in system field SY- MANDT.
LOGICAL_FILENAME	Logical file name that is to be used for validation (uppercase is required).
OPERATING_SYSTEM	Specifies the operating system for which the file name is to be determined. The default value is the operating system of the application server, as specified in system field SY OPSYS.
PARAMETER_1 PARAMETER_2	Can pass any values to the placeholders labeled <PARAM_1> to <PARAM_3> in the physical file and path names.

PARAMETER_3	
WITH_FILE_EXTENSION	Indicates that the file format defined for the logical file name is to be attached to the physical file name as a file name extension.
USE_BUFFER	Indicates that the control tables are to be buffered.
ELIMINATE_BLANKS	Specifies whether all spaces are to be removed from the returned file name.

The import parameters are passed by FILE_VALIDATE_NAME to FILE_GET_NAME, to determine a physical file name. The system compares this to the content of the CHANGING parameter PHYSICAL_FILENAME. You therefore influence the determination of the physical file name that is used for validation.

Export Parameter	**Function**
VALIDATION_ACTIVE	If the value is not SPACE, the system could not determine a physical file name for the logical file name or an assigned alias.
TS_ALIAS	Sorted table of the defined aliases (including those for the passed logical file name).

j. CHANGING Parameter	k. Function
l. PHYSICAL_FILENAME	m. Describes the physical file name that FILE_VALIDATE_NAME compares with the settings of the transferred logical file name. Since the physical file name might need to be normalized, it has the type CHANGING.

n. EXCEPTIONS	o. Function
p. LOGICAL_FILENAME_NOT_FOUND	q. Triggered if the function module FILE_GET_NAME has propagated the exception FILE_NOT_FOUND to FILE_VALIDATE_NAME.
r. VALIDATION_FAILED	s. Triggered if the physical file name in PHYSICAL_FILENAME does not correspond to the restrictions in LOGICAL_FILENAME.

Defining and Using Aliases

To allow the reuse of logical file names when validating file names and to make validation more flexible, you can define aliases for logical file names. An alias is itself a logical file name.

Example for Checking a Physical File Name

The user enters the physical file name **`/usr/sap/PRD_100/tmp/my_file`** in the program ACCESS_APPLSERVER_FILE. Program ACCESS_APPLSERVER_FILE uses the logical file name PHYS_FILE_VALIDATION for validation, which is not assigned to a physical path in delivery status. You already have another logical file name Z_FILE_TMP in your system that generates the physical file name **`/usr/sap/PRD_100/tmp/my_file`**. Instead of generating this assignment for the logical file name PHYS_FILE_VALIDATION again, enter Z_FILE_TMP as an alias for PHYS_FILE_VALIDATION.

Example for Checking Physical File Paths

The user specifies a physical file name in the directory area **`/usr/sap/PRD_100/tmp/`** or **`/usr/sap/PRD_100/work/`** in the program ACCESS_APPLSERVER_FILE2. The program ACCESS_APPLSERVER_FILE2 uses the logical file name PHYS_FILE_VALIDATION2 for validation, which is not assigned to a physical path in delivery status. You define two new logical file names, Z_DIR_TMP and Z_DIR_WORK, that have the data format DIR, and which are assigned the directory paths named above. You enter **`Z_DIR_TMP`** and **`Z_DIR_WORK`** as aliases for PHYS_FILE_VALIDATION2.

Example for Specifying a Logical File Name to the User Interface

The user is only allowed to enter certain logical file names in the program ACCESS_APPLSERVER_LOGFILE. The program ACCESS_APPLSERVER_LOGFILE uses the logical file name LOG_FILE_VALIDATION for validation, which is not assigned a physical file name in delivery status.

You have already defined the logical file names ZACCESS_APPLSERVER_LOGFILE1 and ZACCESS_APPLSERVER_LOGFILE2 in your system, which create valid physical file names. You now enter the logical file names ZACCESS_APPLSERVER_LOGFILE1 and ZACCESS_APPLSERVER_LOGFILE2 as aliases for LOG_FILE_VALIDATION.

The user can now enter the logical file names ZACCESS_APPLSERVER_LOGFILE1 and ZACCESS_APPLSERVER_LOGFILE2 for the program ACCESS_APPLSERVER_LOGFILE. The logical file name LOG_FILE_VALIDATION cannot be entered in this example because no physical file name has been assigned to this logical file name. However, if the user is to be able to use this logical file name, then you have to assign a physical file name to the logical file name LOG_FILE_VALIDATION.

When checking a logical file name it does not make any sense to assign a logical file name with the data format DIR as an alias. The program needs a logical file name that defines a complete physical file name.

Generalization

If you define an alias for a logical file name, the alias is handled in the same way during the validation of a file name as the delivered logical file name of an application for validation. That is, a file name is valid according to validation if it meets the requirements of the original logical file name or a defined alias.

Maintaining Aliases

You can maintain aliases in the Implementation Guide (IMG) by choosing *SAP NetWeaver* → *Application Server* → *System Administration* → *Platform-Independent Filenames* → *Maintain Aliases for Logical File Names*.

This IMG path is dependent on the release and Support Package status of your system and may not exist. If this is the case, you can maintain aliases using transaction **SM30** using the view **V_FILEALIA** (as of release 4.6C: view **V_FILEA31I**).

Using Logical File Names in Programs

To use logical file names in one of your application programs, perform the following steps:

1. Use transaction FILE to check whether definitions are stored in the system for the logical file names. These definitions convert the logical file names to platform-specific file names. If necessary, create new definitions with transaction FILE.

   For more information, see the Implementation Guide under *SAP NetWeaver* → *Application Server* → *System Administration* → *Platform-Independent File Names*.

2. Ensure that the physical paths specified in these definitions exist in the runtime environment file system. If necessary, create the required directories, or contact your system administration.

3. Test the file name conversion by calling the function module in the individual test of transaction SE37.

   When performing the individual test in transaction SE37, set the Uppercase/Lowercase indicator, since the field OPERATING_SYSTEM, in particular, is case-sensitive. Correspondingly, you need to specify the logical file name entirely in uppercase characters.

4. Add a call of the function module to your program. To do this, you can choose *Edit* → *Pattern* in the ABAP Editor.

For more information about saving files on the application and presentation servers, see *Using Files* in the *ABAP User Guide*.

Note that creating a syntactically correct logical file name does not in itself guarantee the successful saving of a file. The physical path must also actually exist in the file system at runtime.

Do not allow the user to enter any available logical file name on the user interface during execution. Store the logical file name in the program logic, or follow the instructions for using logical file names in the user interface. For more information, see Secure Programming - ABAP [Extern].