SMARTdata UTILITIES for Windows

Distributed FileManager User's Guide

SC26-7134-00

SMARTdata UTILITIES for Windows

Distributed FileManager User's Guide

SC26-7134-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page v.

First Edition (April 1997)

This edition applies to the SMARTdata UTILITIES, Version 2 Release 1, and to all subsequent releases and modifications until otherwise indicated in new editions. SMARTdata UTILITIES is shipped with IBM VisualAge for COBOL. Make sure you are using the correct edition for the level of the product.

Publications are *not* stocked at the address below. Requests for IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

You can order by calling IBM Software Manufacturing Solutions at 1-800-879-2755.

A form for reader comments is provided at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
RCF Processing Department
G26/050
5600 Cottle Road
SAN JOSE, CA 95193-0001
U.S.A.

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to :

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. M13
5600 Cottle Road
San Jose, CA 95193

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and service marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

IBM	OS/2
MVS/ESA	OS/400
Operating System/2	VM/ESA
Operating System/400	

The following terms are trademarks of other companies:

Windows	Microsoft Corp.
Windows NT	Microsoft Corp.
Windows 95	Microsoft Corp.

About this book

This publication helps the application programmer use the Distributed FileManager (DFM) for Windows program.

This publication describes how to use DFM for remote record access. Syntax diagrams show you how to code the keywords to set up the configuration file, perform data conversion, and how to enter the user interface commands. DFM uses CDRA coded character set identifiers (CCSIDs) to define the language for data conversion.

DFM uses Data Description and Conversion (DD&C) for Windows to perform data conversion.

To work with Data Description and Conversion for Windows, you should be familiar with the following:

- The Windows operating system
- C programming language
- Any other programming languages that you might use when converting data, such as COBOL or PL/1, including their respective compilers and operating systems
- IBM's A Data Language (ADL), which describes the encodings of data types and structures.
- Character Data Representation Architecture (CDRA).

Bibliography

You can order books by calling IBM Software Manufacturing Solutions at 1-800-879-2755.

Table 1. SMARTdata UTILITIES for Windows Publications

Publication Title	Order Number
SMARTdata UTILITIES for Windows Set	SBOF-6135
SMARTdata UTILITIES for Windows Distributed FileManager User's Guide	SC26-7134
SMARTdata UTILITIES Data Description and Conversion	SC26-7091
SMARTdata UTILITIES VSAM Application Programming Interface Reference	SC26-7133
SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion	SC26-7092

Table 2. Other Publications

Publication Title	Order Number
DDM Architecture: Specifications for ADL	SC21-8286
Character Data Representation Architecture, Level 2	SC09-1390
IBM Systems Journal: Volume 31, No. 3, 1992	G321-5483
IBM Dictionary of Computing	SC20-1699
Compilers—Principles, Techniques, and Tools: by the Addison—Wesley Publishing Company	
IEEE Standard for Binary Floating—Point Arithmetic: ANSI/IEEE STANDARD	754-1985
INTEL 387™ DX User	

Using Syntax Diagrams

This section describes how to read the syntax diagrams provided for each command in this guide.

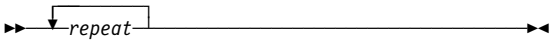
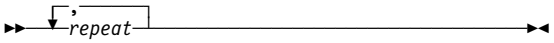
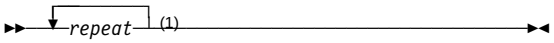
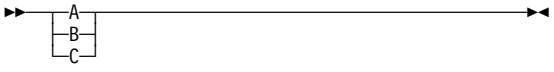
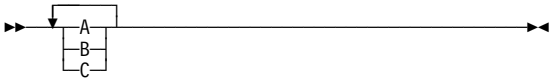
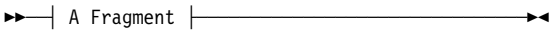
To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►— symbol indicates the beginning of a syntax diagram.
- The —► symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ►— symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The —►◄ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional).

Syntax Diagram Description	Example
Abbreviations: Uppercase letters denote the shortest acceptable abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated. You can type the item in uppercase letters, lowercase letters, or any combination. In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.	►►—KEYWOrd———————————————————►◄
Symbols: You must code these symbols exactly as they appear in the syntax diagram.	* Asterisk : Colon , Comma = Equal Sign - Hyphen () Parentheses . Period
Variables: Highlighted lowercase items (*like this*) denote variables. In this example, *var_name* represents a variable you must specify when you code the KEYWORD command.	►►—KEYWOrd—*var_name*———————————————►◄

Syntax Diagram Description	Example
Repetition: An arrow returning to the left means that the item can be repeated. A character within the arrow means you must separate repeated items with that character. A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.	   **Note:** 1 Specify *repeat* up to 5 times.
Required Choices: When two or more items are in a stack and one of them is on the line, you *must* specify one item. In this example, you must choose A, B, or C.	
Optional Choice: When an item is below the line, the item is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 
Defaults: Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C. You can also specify the default explicitly.	
Repeatable Choices: A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragments: Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named “A Fragment.”	 **A Fragment:** 

Chapter 1. Overview of Distributed FileManager for Windows

The Distributed FileManager for Windows (DFM) offers client services for creating, reading, and writing record-oriented data on other computing systems.

DFM is an implementation of a source (client) system as defined by the Distributed Data Management Architecture (DDM) and operates according to the prescribed rules and protocols with DDM target (server) systems.

The objective of this section is to describe what DFM can do, what it takes to use it, and the basics of how it works.

Note: The **local VSAM file system** component of SMARTdata UTILITIES is **not** available on the Windows platform.

Terminology

The following terms are used in this chapter and subsequent DFM discussions in this document. Some of the terms are typically used in client/server environments and can be very familiar to the reader. Some of the terms have particular usages specific to DFM.

Distributed Data Management Architecture
Distributed Data Management Architecture (DDM) offers a vocabulary and set of rules for sharing and accessing data among like and unlike computer systems. DDM includes a set of standardized file models for direct, keyed, relative, and sequential record data as well as byte-stream data. It allows users and applications to access data without concern for the location or format of the data.

The DDM architecture refers to the connected systems as **source** and **target**. These terms correspond to the more widely recognized **client** and **server** models. Consequently, in this document, the terms source and target are used interchangeably with client and server.

Distributed FileManager for Windows (DFM)
Distributed FileManager for Windows is an implementation of source (client) support as specified by DDM. DDM permits systems in a distributed environment that have DDM source (client) capability to access file data on a DDM target (server) system.

local Local is your reference point when discussing such entities as files, platforms, or applications. For example, a local file is a file that resides on the same system as the one that requested access to the file data.

platform Platform refers to a computer system running a specific operating system and connected to a network. For example, Windows NT, Windows 95, OS/2, OS/400, and MVS/ESA are different operating system platforms.

record-oriented file
A file with a record-oriented structure that is accessed record by record. This file structure is typical of data sets on VM, MVS, and OS/400 systems. Contrast with *stream-oriented file*.

remote Remote is relative to your reference point when discussing such entities as files, platforms, or applications. For example, a remote file is a file that resides on some system other than the system that initiated the request for data.

source Source is the term used in DDM to refer to the platform that originates a request for remote data. Source is also known as the **client**. Source and client are used interchangeably within the scope of this document. Contrast with **target**.

stream-oriented file
A file with a byte-oriented structure that is accessed as continuous streams of data bytes. This file structure is common in workstation environments. This is also known as a byte-stream file. Contrast with *record-oriented file*.

target Target is the term used in DDM to refer to the platform that fulfills a request for remote data. Target is also known as the **server**. Target and server are used interchangeably in this document. Contrast with **source**.

Functional Highlights

DFM for Windows implements:

- **Data access to remote, record-oriented files**

  DFM for Windows supports:

  – The record types defined by the DDM architecture: sequential, direct, keyed, and alternate index.

  – Access to remote files on server systems which have a DDM target implementation.

  The DDM target implementation maps the DDM record types above into the appropriate file models on the server. For example, the DFM component of DFSMS/MVS maps DDM record types into mainframe SAM, VSAM, PDSE, and PDS data sets.

  DFM for Windows does **not**:

  – Provide access to local files.
  – Provide support for stream-oriented (byte-stream) files.

- **Data Description and Conversion (DD&C)**

  DFM for Windows can optionally provide for data conversion of record-oriented files. A typical application of this capability is when a user wants to view a remote file on his or her system and the file is stored in EBCDIC on the server system. In this case, the user most likely needs to convert the data to ASCII. The user can

encode information using IBM's A Data Language (ADL) to describe this conversion.

- **Application Program Interfaces for Remote File Access**

  Remote file access by DFM for Windows is invoked when an executing Windows application calls a DFM application programming interface (API). These APIs are called Virtual Storage Access Method (VSAM) APIs, or simply, VSAM APIs.

  The fact that a file is “remote” is indicated when the API is called. The API call also indicates an encoded specification of a particular remote server. The file name parameter in a VSAM API call actually has extra characters immediately preceding the remote file name. These characters include:

  1. Two back slashes (\\) at the start of the parameter.
  2. Some more characters that end with another back slash. These characters comprise either the **machine name** for the remote server or a **shortcut name** (described in the next item).

  This technique is very similar to the Windows UNC (Universal or Uniform Naming Convention) technique which is used for network file access.

  A user running a Windows application will be unaware of the VSAM API invocation. Likewise, an application developer using a high level language (HLL) such as COBOL or PL/I will also be unaware of the VSAM API invocation. This is because the HLL runtime library calls the VSAM APIs appropriately to handle the access to a remote record file. For those application developers who wish to work in C and invoke DFM from Windows, the *SMARTdata UTILITIES VSAM Application Programming Interface Reference* provides information on the VSAM APIs.

  If the file name can be dynamically changed using **environment variables** set at runtime, the application developer does not have to be concerned with the location of the data. For example, assume a COBOL application contains the statements:

  ```
  FILE-CONTROL.
       SELECT FILE1 ASSIGN TO sysin
  ```

  This statement refers to the environment variable *sysin*. The user of the application, at run time, can define the particular file to access by issuing the **set** command for this environment variable. If the file resides on the user's workstation, the **set** command might look like this:

  ```
  set sysin=d:\userdir\myfile
  ```

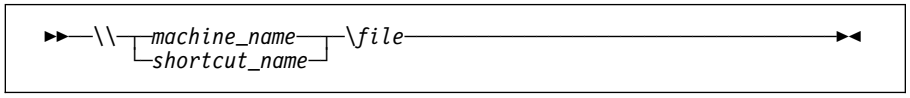
  If this file is a remote file on a server with a machine name of MVS2, the set command might look like this:

  ```
  set sysin=\\MVS2\user.myfile
  ```

  where *user.myfile* is the remote file's name on the MVS2 system.

- **Specification of remote resources**

  As mentioned above, the dynamic specification of a remote file for DFM access is indicated in the usage of the UNC style format. The UNC format is:



where:

machine_name
Represents either the SNA LU alias name as specified in the SNA configuration or the SNA fully-qualified network address of the remote server system.

shortcut_name
Represents an alias for the remote server system and optional file name prefix for the remote file.

file Represents the name of the remote file to access on the server system

An application can use a “shortcut name” at the front of its references to remote file names. For example, in “\\IBM\FILE,” the shortcut name is “IBM.” At run time, you can equate the shortcut name to the proper machine name (server) and optionally specify one or more qualifiers to the remote file name. If you equate “\\IBM” to “\\MVSESA\IBMUSER.PART1,” then DFM recognizes the machine name to be “MVSESA” and the remote file name to be “IBMUSER.PART1.FILE.”

- **A line command user interface**

  Some *setup* information must be provided to DFM in preparation to dynamically access remote data. For example, a user typically has to provide a logon id and password for access into the particular server system. Likewise, the user can tailor special network access requirements, such as “mode name” to the particular server.

  If an application uses shortcut names in its remote file name references, the user needs to assign the correct machine name and possibly a file prefix to each shortcut name.

  DFM provides three line commands to address this “setup” information: *dfmlogon*, *dfmcfg*, and *dfmnet*. One or more of these commands is typically run before performing remote file access.

- **Tailoring of DFM Information**

  Two DFM line commands produce output control files to store the “setup” information. This information can be defined once and reused many times. In fact, one user might build common server system and data conversion definitions, and then propagate the resulting control files across other workstations.

  As a convenience, DFM also applies default values for server system parameters where no values have been defined in the user's control files.

- **System Administrator Controls**

  The DFM line commands and initialization routines are designed to recognize and concatenate two sources of control information:

- System-wide or “global” control information, typically defined by a system administrator who gives other users the authority only to read the information, and
- Private or “local” control information, which the individual user creates and stores on his or her workstation.

It is not mandatory to provide control information at both the system-wide and private levels. The system-wide facility is provided for the convenience of an installation which has network sharing capabilities and might prefer to centrally administer network connection details. Likewise, the private control information facility is provided in case there is no network sharing capability or a user would like to augment or override the system-wide control information.

The DFM line commands and DFM's remote file processing work in conjunction with two environment variables called DFMUSER and DFMADMN. These indicate where the directories for DFM's private and global control files, respectively, are located.

- **Conversation Handling Strategy**

  DFM allocates one conversation to a server for each Windows application process.

- **Single Thread Processing**

  DFM does not support the execution of multiple threads within a single process. The behavior of DFM when called by more than one thread within a single process is unpredictable.

- **Supported DDM Level**

  The DDM architecture has several *levels* or versions published. DFM for Windows supports Distributed Data Management Level 4.0 and the duplicate key enhancement Level 5.0. DFM for Windows interacts with “back level” DDM servers (ones which have implemented DDM Levels 1 through 4).

Requirements

DFM for Windows uses SNA protocols to communicate with server systems. This SNA support is not part of SMARTdata UTILITIES; there are SNA network software packages available for this purpose. One of these network software packages has to be installed on a Windows system before invoking DFM for Windows.

Operating Systems

DFM runs under the control of:

- Windows 95
- Windows NT 3.51

Programs Required

DFM for Windows requires a network access software package which has a CPIC interface. Some of the possibilities include (you only need one):

- IBM Personal Communications or
- Microsoft SNA Server

Some examples of DDM server counterparts are:

- DFSMS/MVS version 1.2.0 or later (for MVS)
- OS/400 DDM support (for AS/400)

Migration and Coexistence

It is not necessary to upgrade back-level DDM servers to DDM level 4 in order to communicate with DFM. DFM sends commands, parameters, and values that can be handled by a particular DDM server. If a DDM server cannot support a requested DDM command, the server will return an error.

How DFM is Connected to Server Systems

Figure 1 illustrates how Windows applications using DFM logically access remote files in various server systems. For example, a Windows application needs to access a file "MYID.FILE1" on the MVS system. The MVS system has been defined to the netware access software as LU alias MVS1. The application can execute a VSAM API to open the file with a file name parameter value of "\\MVS1\MYID.FILE1."

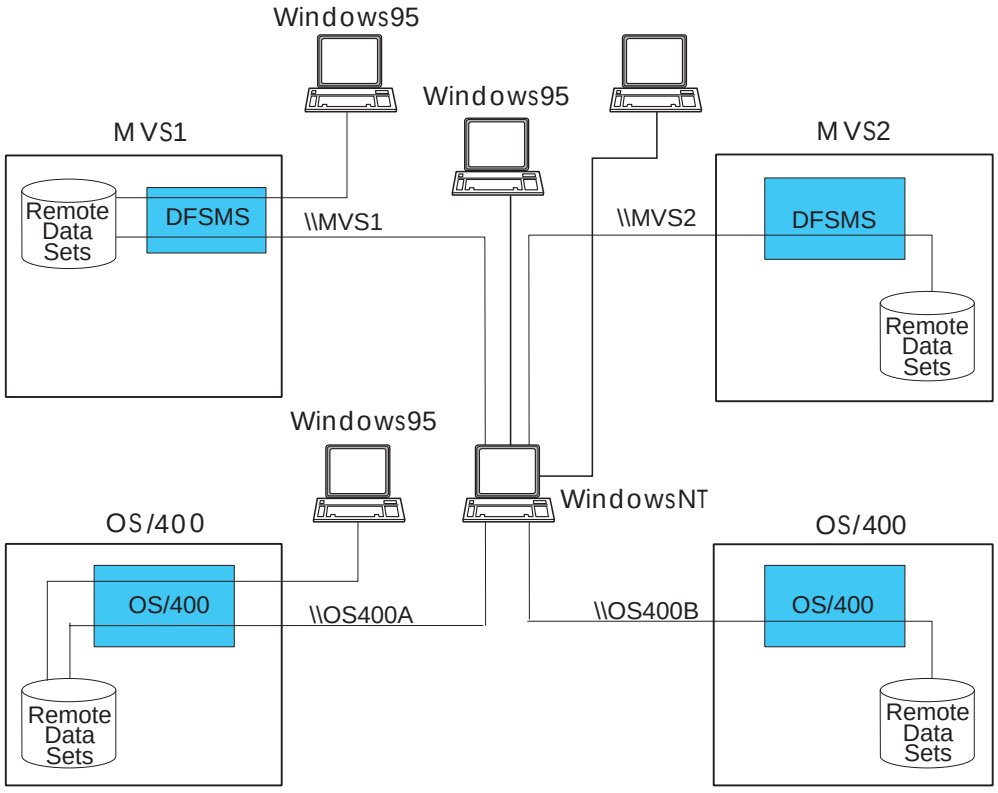


Figure 1. DFM Servers Accessed by DFM for Windows

How to Get Started

Before performing the setup steps, you will need to consider certain items of information.

- The LU alias or the SNA fully-qualified network address of each server system to be accessed.
- The level of security required for each server system.
- The SNA “mode” for each server system (the possible modes are defined in the network access software package).
- The maximum number of bytes that can be sent to each server system at one time.
- User IDs and passwords to be used to establish communication with each server system.
- How to activate the network software.

Before you can use DFM to access files through a Windows application program, you must:

1. Start the network software and activate the network link from the workstation.

   DFM uses the SNA LU 6.2 protocol to communicate with server systems. The LU 6.2 protocol uses the Advanced Program-to-Program Communication (APPC).

   You will need to configure your network access software for your SNA network.

   Note: The SNA network should be set up so that the workstation that uses DFM is configured as an **independent LU** capable of parallel sessions.

2. Set up the DFM text configuration files with the parameters required by DFM to establish communication and perform data conversion for remote files.

   A good starting point for this step is to locate the directory:

   ```
   <nonvis>\samples
   ```

   where *<nonvis>* represents the “nonvisual tools” directory of the product in which DFM is shipped. This directory contains a sample DFM configuration file that can be customized.

   The system administrator can optionally set up a system-wide text configuration file.

   Individual users can set up private text configuration files.

   The configuration file is discussed in Chapter 2, “The Configuration File” on page 11.

3. Check that the DFMADMN (if system-wide data is needed) and DFMUSER environmental variables point to the correct directories for global and private DFM setup information.

   Execute the *dfmcfg* command to create a binary configuration file. Specify (as the input file) the name of the text configuration file created in the previous step.

4. Optionally execute the *dfmnet* command. This command allows users to define "shortcut" names for:
   - Server systems
   - Prefixes to files on server systems

   This command is provided as a convenience for users who want to identify server systems and sets of remote files in their own customized manner. This command is discussed in Chapter 3, "DFM Commands" on page 19.
5. Execute the *dfmlogon* command. This associates the user's remote user ID and password with a server system. DFM uses this information when allocating a conversation with the server system.
6. Start the application.

How DFM for Windows Works

Figure 2 on page 9 helps to illustrate some basic steps involved, and the role DFM for Windows plays in accessing remote files. The casual user of Windows might not need to study this section. However, this might help the person who has to provide detailed setup information for DFM to get an overview of remote file access and the reasons that certain information is needed by DFM.

The server system is where the file resides. The client system contains the Windows application that is accessing the remote file.

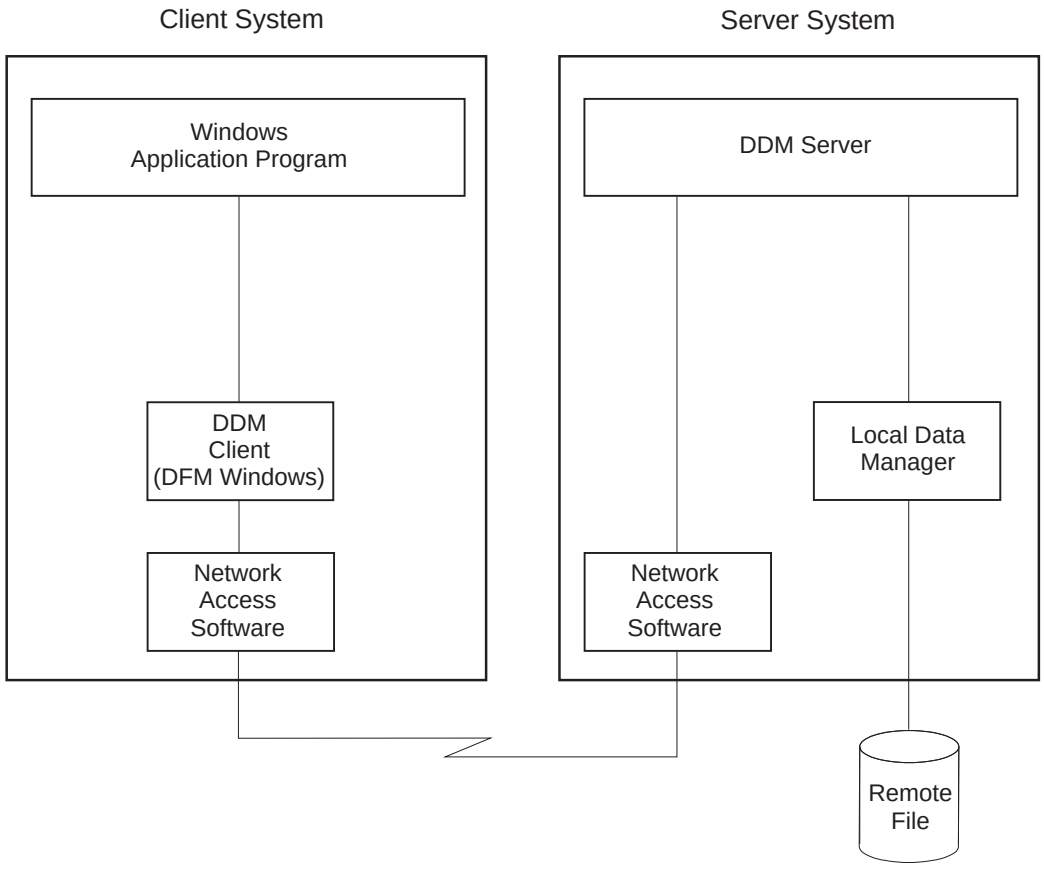


Figure 2. Overview of DFM Processing

The steps involved in accessing a remote file are:

1. An application program issues a VSAM API request directly or uses an HLL file access command. The HLL file access command triggers the runtime library to issue a VSAM API request.
2. DFM for Windows processes the request and passes it to the network access software.
3. The network access software transmits the request to the server system.
4. The server system's communications manager receives the request and forwards it to the DDM server.
5. The DDM server processes the request for remote data.
6. The Local Data Manager is invoked to retrieve the requested data and to send it back to the DDM server.
7. The DDM server builds a reply “data stream” and transmits it back to the client system through its network access software.

8. DFM for Windows is notified by the network access software that it has a reply. DFM analyzes the reply and passes the results back to the application.

Note: If an error occurs in these steps, a DFM Reply Message is returned by the software that detects the error. Examples are: a common link failure, server security failure, or a remote file is locked.

Chapter 2. The Configuration File

DFM uses the configuration file statements to establish communication between client and server systems. The configuration file statements define two aspects of communication:

Conversation control This names the server system, and defines the buffer size, the mode name, and the security information for communication between the client system and the server system.

Conversion control This is optional. This names the server system; each file on the server system to undergo data conversion, and the name of a conversion file to be used by DD&C to perform the conversion.

The rest of this chapter describes the text file that a user can create to define these statements. The *dfmcfg* line command is later used to create a binary or "formatted" configuration file from this text file.

Note: You might not have to build a text configuration file from scratch. There might be a suitable file available in your installation. If not, consider referring to the sample input text configuration file in Appendix A, "Distributed FileManager Configuration Examples" on page 63.

General Instructions for Creating a Configuration File

Use the following guidelines when creating or editing a configuration file.

- Create the file with a text editor (for example, an MS or DOS text editor). DFM ignores spaces, tabs and new line characters.
- The text configuration file is composed of *statements* which contain *parameters.* Each statement must end with a semicolon.
- Comments are allowed between statements. Delineate a comment by a slash and an asterisk (/*) at the beginning of the comment, and an asterisk and a slash (*/) at the end. For example:

```
/* This is a comment. */
```

- Some parameters are order-dependent; these will be identified in their descriptions.
- Unless specified otherwise, parameters must be in upper case. The conversation parameters are **case-sensitive**. These parameters, as well as the remote file names, are transmitted to a server system as specified in the text configuration file.
- The suggested name for a text configuration file is *dfm.rc*. This is the name used in examples in this document.

Conversation Control

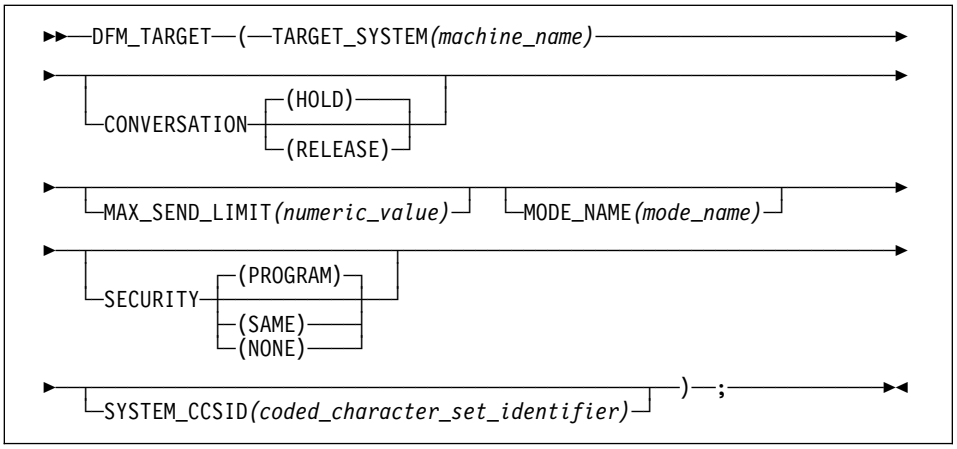
The statements below define the rules and parameters that DFM uses when establishing a conversation with a server system.

DFM_TARGET Statement

There should be one DFM_TARGET statement for each server system that requires conversation parameters other than default values. See “Conversation Control Default Statements” on page 14. Values in a DFM_TARGET statement override the default values.

DFM_TARGET Syntax

This is the syntax for the DFM_TARGET statement.



Parameters

TARGET_SYSTEM(*machine_name***)**

This is the only required parameter. It must be the first parameter specified. You can specify either:

- The logical unit alias (or LU alias) of the server system, as defined in the network software's configuration. This name must consist of 1 to 8 characters.
- The SNA fully qualified network address of the server system. There is a specific naming convention for such names which consists of:
  1. 1 to 8 characters (network name)
  2. Followed by a period
  3. Followed by 1 to 8 characters (which is the server LU name)

  Note: Some network access software configurations are case sensitive. Be sure to match the “machine name” case to that of the network access software configuration.

CONVERSATION(HOLD|RELEASE)

This specifies the duration of the LU 6.2 conversation that DFM uses when communicating with the server system.

HOLD DFM deallocates the conversation upon the termination of the process in which DFM is executing. The HOLD option reduces the response time of the server system in some situations. This is the default.

RELEASE DFM deallocates the conversation when the last file on the server system has been closed. The RELEASE parameter minimizes the time an application is connected to a server system.

MAX_SEND_LIMIT(*numeric_value***)**

Specifies, in bytes, the maximum buffer size that you can send to the server system. The value is an integer between 256 and 32767, inclusive.

Note: No DFM object, a record for example, which has a size larger than the value specified on the MAX_SEND_LIMIT parameter can be sent by DFM to a partner LU or received by DFM from a partner LU. An error message is returned by DFM to the calling application if an attempt is made to send an object that is larger than MAX_SEND_LIMIT. It is recommended that the MAX_SEND_LIMIT value be at least 16 bytes larger than the length of the largest record you want to process.

The default is 4096 bytes.

MODE_NAME(*mode_name***)**

Allowable mode names are defined by the network access software. The name must consist of 1 to 8 characters. Be sure the use of upper and lower case alphabetic characters in this parameter is correct. The default is #INTER.

SECURITY(PROGRAM|SAME|NONE)

Specifies the security DFM uses when establishing an SNA conversation. See your network access software configuration guide for more information.

PROGRAM
Indicates that a user ID and password are included in the allocation request. This is the highest security level.

SAME Indicates that the current program running under the local workstation has been verified for user ID and password. This security information is passed along with an “already-verified” indication.

NONE Indicates that no access security information is available.

The default is PROGRAM, the highest security level.

SYSTEM_CCSID(*coded_character_set_identifier***)**

This is the name of the CDRA coded character set identifier (CCSID) to be used in character conversion for the server system.

This parameter has effect only when there is a FILE_DESCRIPTOR_MAP keyword containing a BASE_ADL_FILE parameter whose ADL statement specifies a CCSID

of 0 for this file. Any other CCSID value in the BASE_ADL_FILE overrides the SYSTEM_CCSID parameter.

The default for this parameter is Latin-1 (CCSID 500), which stands for International EBCDIC. For other CCSID values supported by CDRA, see *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion*.

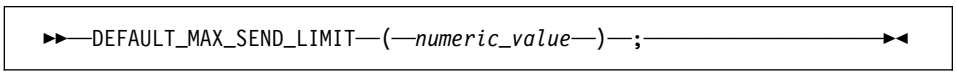
Conversation Control Default Statements

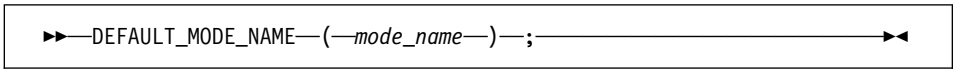
System administrators and end users can define defaults for the DFM_TARGET statement parameters:

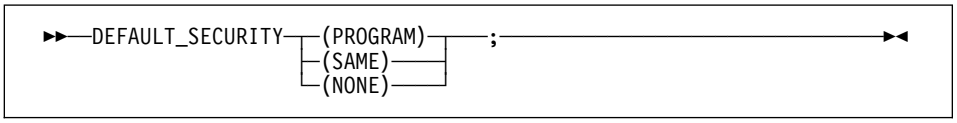
- MAX_SEND_LIMIT
- MODE_NAME
- SECURITY

These statements should be used only if you want defaults different from the standard defaults for these parameters.

Note: These default statements must be specified **before** any DFM_TARGET statements in the text configuration file.







See the parameter descriptions described in “Parameters” on page 12.

Data Conversion Control Statements

The FILE_DESCRIPTOR_MAP statement of the configuration file controls the data conversion for remote record file data. It can specify one of two types of data conversion:

- Text Conversion

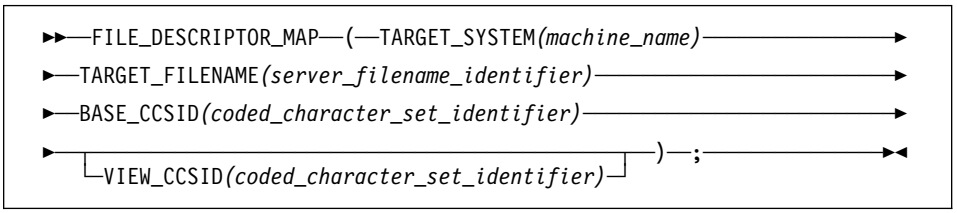
  During text conversion, the data in the remote file is converted from the remote file's character set into the character set of the local client system.

- Complex Data Conversion

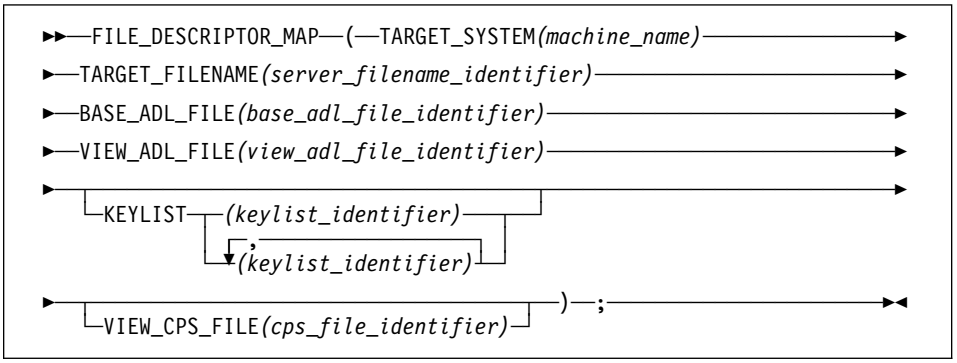
  Complex data conversion translates records that contain a mixture of text and numeric fields. The conversion uses ADL descriptions of the base and view formats to translate the data. Refer to Chapter 4, “Converting Record File Data” on page 27 for information on creating an ADL description file.

Data conversion for any file is optional.

Syntax for Text Conversion



Syntax for Complex Data Conversion



The *identifier* value used in the above parameters has the following properties:.

- It can contain any valid character in the local environment character set, except the single quote.
- If the remote file name requires parentheses (as in an MVS PDSE member) or white space characters, then enclose the file name in single quotes (for example: 'dfm.pdse(mbr1)').

Parameters Required for All Conversions

TARGET_SYSTEM(*machine_name***)**

Specifies the LU alias or fully qualified network address of a server system. The machine_name must match the TARGET_SYSTEM parameter value of a DFM_TARGET entry that has already been defined. Be sure to match the case of the “machine name” to that in the network access configuration.

TARGET_FILENAME

Specifies the name of the remote file for conversion.

DFM interprets “*” and “?” characters in the remote file name as wildcard characters. See“Pattern Matching Expressions” on page 17 .

DFM converts the TARGET_FILENAME identifier into CCSID 0500 (or EBCDIC characters) of the server system when accessing remote files.

Parameters Only for Text Conversion

BASE_CCSID

Specifies the CCSID of the text data in the remote file. This parameter is required for text data conversion.

VIEW_CCSID

Specifies the CCSID of the text data on the local file system. This parameter is optional for text data conversion. If it is not specified, the CCSID for the active OEM code page (also known as the DOS code page) code page on the client Windows system is used.

Parameters Only for Complex Data Conversion

BASE_ADL_FILE

Specifies the fully qualified name of the local file containing the ADL description of the remote file. We recommend ADL file names end in the suffix '.adl', but this is not required. During complex data conversion, the file data is converted from the format described in this file into the format described by the file named in the VIEW_ADL_FILE parameter.

This parameter is required for complex data conversion.

VIEW_ADL_FILE

Specifies the fully qualified name of the local file containing the ADL description of the desired view of the remote file. We recommend ADL file names end in the suffix '.adl', but this is not required. During complex data conversion, the file data is converted from the format described in the BASE_ADL_FILE into the format described by the file named in this parameter.

This parameter is required for complex data conversion.

KEYLIST

A list of one or more uniquely qualified identifiers that match the name of the key field(s) within the base and view ADL declarations. Multiple key identifiers must be separated by commas. Refer to section "Creating ADL Descriptions for Keyed Files" on page 29 for more information on providing key names. Be sure to match the "key identifier" case to that of the key field(s) in the ADL declarations.

This parameter is required for keyed files only.

VIEW_CPS_FILE

Specifies the fully qualified name of the local binary file that is created for complex data conversion.

The VIEW_CPS_FILE parameter is optional. If it is not specified, then DFM gives the file the same name as the view ADL file with the "*.adl*" extension replaced by "*.cps*", or "*.key.cps*" if the file is keyed. If the view ADL file does not end in "*.adl*" then *dfmcfg* appends the "*.cps*", or "*.key.cps*" to the view ADL file name.

Pattern Matching Expressions

Sometimes it is convenient to use pattern matching expressions when you aren't sure of the name of a file, or you want to list several files whose names are similar.

Pattern matching expressions are created by using wildcard characters.

- An asterisk (*) matches zero or more characters in a particular position in a name.
- A question mark (?) matches a single character in a particular position in a name.

Examples using wildcard characters:

*****	all names
***nam**	matches abcnam or cdenam
***.*nam**	matches gdm.winnam or abc.attnam
.nam	matches dos.namjohn
ab?.nam	matches abc.nam

Chapter 3. DFM Commands

This chapter describes the commands associated with DFM for Windows:

dfmcfg Creates the DFM binary configuration file.

dfmnet Adds or removes a shortcut name from DFM's list of shortcut names (optional).

dfmlogon Adds, modifies, or deletes a user ID and password for a server system

Notes Applicable to All DFM Commands

These notes are applicable to all DFM commands.

- Each command issues a title message, unless you explicitly request that the message be suppressed. The message is:

  ```
  Licensed Materials - Property of IBM
  Distributed FileManager for Windows
  5639-B92 (c) Copyright IBM Corp. 1997.  All rights reserved.
  Version 2
  ```

  The flag to suppress this message is **-q**.

- Flags for each command and parameters naming local files, can be specified in upper or lower case.
- DFM passes some user-specified parameters directly to other software. For such parameters, DFM does not change any character that can have an upper or lower case. However, the recipient software, such as the network access software, might or might not change the case. Thus, depending on the network access software installed, “mvs1” might or might not be changed to “MVS1.” We will note in each parameter description where you need to specify the case correctly.
- Two environment variables, DFMADMN and DFMUSER, are accessed by the commands. In general:
  - The DFMUSER environment variable can be set to a directory that is located on the user's workstation. This is known as a “private” directory.

    If DFMUSER is not set, a DFM command assumes the private directory is the current directory. The current directory is the drive and directory of the MS-DOS Window in which the command is being issued.
  - The DFMADMN environment variable can be set to a directory that is shared across many Windows workstations. This is known as the “global” directory and is optional.

    If DFMADMN is not specified, a DFM command determines whether “global” or “private” information is appropriate, based on the settings in the command. Unless certain settings are explicitly made, it is assumed the DFM information is for the user's “private” directory.

You can set these environment variables in their current MS-DOS window by issuing the MS-DOS *set* command. For example:

```
SET DFMUSER=D:\MYDIR
```

sets the DFMUSER environment variable to the MYDIR directory on your D drive.

Note: The command:

```
SET DFMUSER=
```

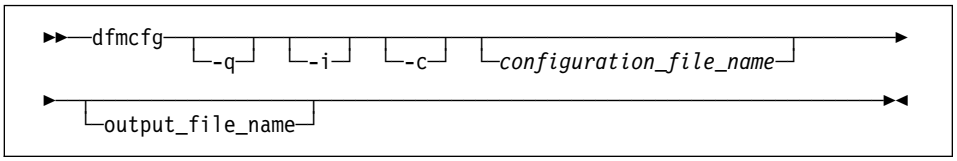
sets the DFMUSER environment variable to an undefined value.

You can also establish DFMUSER and/or DFMADMN environment variables for **all** the user windows. See the appropriate Windows manual for specific information.

dfmcfg Command

The *dfmcfg* command reads a text configuration file and creates a binary or formatted configuration file. The binary file contains the server system parameters DFM uses during remote file access. The binary file's name is *dfmcfg.dfm*.

Syntax



Flags

-q Use this flag when you do **not** want to display the title message.

-i Use this flag when you do **not** want to inherit the settings of the global configuration file (the configuration file in the directory defined by the DFMADMN environment variable).

-c Use this flag to display the active binary configuration file in readable text form. Neither the **-i** flag nor any other parameters are permitted in conjunction with the **-c** flag.

Parameters

configuration_file_name

This is the path-name of the input text configuration file. This parameter is required unless the -c flag is specified.

output_file_name

This is the path-name of the output formatted configuration file. This parameter is optional. If it is not specified and an output file is created, the output file will be called **dfmcfg.dfm**.

Remarks

This section provides additional information about the command.

Configuration File: If you do **not** include the **-i** flag, DFM attempts to find a global binary configuration file (using the DFMADMN environment variable) and read this file. DFM then modifies the contents with the specified text configuration file, and generates a new binary configuration file.

You can prevent the global configuration file from being included by:

- Using the **-i** (ignore) flag when you enter the *dfmcfg* command. For example:

  ```
  dfmcfg -i input_filename [output_filename]
  ```

- Setting the DFMADMN environment variable to "undefined." For example:

  ```
  set DFMADMN=
  dfmcfg [valid_flag] input_filename [output_filename]
  ```

Placement of Output Binary Configuration File: If the output file name parameter is specified (as a full path name), the new binary configuration file is stored using this path. If this parameter is not specified, the DFMUSER environment variable is the next directory that is selected for storage. If neither DFMUSER nor the output parameter is specified, the binary configuration file is stored in the current directory.

Selection of Active Binary Configuration File: DFM searches for the private binary configuration file first. If the private configuration file is not found, DFM searches for the global binary configuration file. If both the private and global binary configuration files cannot be found, DFM looks for the binary configuration file in the current directory.

Note: The DFMUSER environment variable defines the directory for the private binary configuration file. Similarly, the DFMADMN environment variable defines the directory for the global binary configuration file.

Building a Global Configuration File: Set the DFMADMN environment variable to the path for the directory used for global usage. On the *dfmcfg* command, define the output file name to be the full path name of the global binary configuration file (using the same directory as DFMADMN specifies). For example:

```
set DFMADMN=D:\DFMADMN\GLOBAL
dfmcfg -q input_filename D:\DFMADMN\GLOBAL\dfmcfg.dfm
```

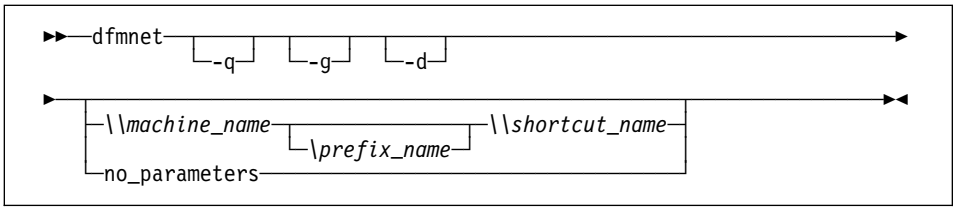
Alternatively, use the ignore flag (**-i**) to build the binary configuration file in any convenient directory. Then, copy the file to the global directory.

dfmnet Command

The *dfmnet* command builds and updates a file comprised of shortcut names. The name of this file is *dfmnet.dfm*.

This command is optional.

Syntax



Flags

-q Use this flag if you do not want to display the title message.

-g Use this flag to make a global shortcut name. For system administrator use only.

-d Use this flag to remove a private or global shortcut name. The **-g** flag must be specified for global removal.

Parameters

(no parameters)
This displays a list of all shortcut names and their associated *machine_name* and *prefix_name* values.

\\shortcut_name
This is the alias name for a server system and optional prefix name for remote files.

\\machine_name
The name of a server system. This name can be in LU alias format (1 to 8 characters) or in SNA network qualified address format (1 to 8 characters, a period, 1 to 8 characters). Eligible names are defined in the network software configuration files accessible to the user's workstation.

Note: The machine_name is case sensitive. Be sure the use of upper and lower case alphabetic characters in this parameter is correct.

\prefix_name
The file name prefix for remote files (optional) on the specified server system.

Note: The prefix_name is case sensitive. Be sure the use of upper and lower case alphabetic characters in this parameter is correct.

When coding the *machine_name, prefix_name*, and *shortcut_name*, be sure you have appropriate spaces as shown below:

```
\\machine_name   \\shortcut_name
\\machine_name\prefix_name  \\shortcut_name
```

Remarks

This section provides additional information about the command.

Using the dfmnet Command: The shortcut names produced by this command can be used for several purposes:

- To associate a server system with another name (the shortcut name). Instances where this might be useful include:
  - When the machine name, such as the LU alias or SNA fully qualified address, is difficult for a Windows application user to remember.

    The user can use *dfmnet* to define their personal shortcut name for the server system. Later, when they are ready to use *dfmlogon* to record the user ID for the server system, they can first list their shortcut names through *dfmnet* to look up the server system's machine name.
  - When a Windows programmer needs to write an application that accesses remote files on various server systems (or the server system is defined after the program is written).

    The programmer can encode the shortcut name in the file name parameter of a VSAM API. Then, when the application is installed and ready to run on a Windows system, the end user can associate the predefined shortcut name with a real server system through the *dfmnet* command.
- To access a specific group of remote files on a server system. If a shortcut name is encoded in the Windows application as described above, the user of the application can equate the shortcut name to a real server system name. Also, the user can define the leading characters for the files to be accessed on this server system. These leading characters constitute a prefix name. The prefix name is essentially one or more high level qualifiers for the remote files on the server system. By repeatedly redefining the shortcut name through *dfmnet*, the user can alter the prefix name and thus selectively access different groups of remote files on the same server system.

The -g Flag and DFM's Environment Variables: If the user specifies the **-g** flag when adding or deleting a shortcut name, the *dfmnet* command modifies the global *dfmnet* control file. The DFMADMN environment variable defines the directory where the global *dfmnet* control file is located. If DFMADMN is undefined or set to a directory that does not exist, an appropriate error message is issued.

Note: The system administrator must have write authority to the global *dfmnet* control file.

If the user does not specify the **-g** flag when adding or deleting a shortcut name, the *dfmnet* command modifies the private *dfmnet* control file. The DFMUSER environment variable defines the directory where the private *dfmnet* control file is located. If the DFMUSER environment variable is set to a directory that does not exist, an appropriate error message is issued. If DFMUSER is undefined, the *dfmnet* command attempts to create or modify the control file in the current directory.

Note: End users have write authority to the private *dfmnet* control file.

Building Global Shortcut Names: This takes two steps:

1. Set the DFMADMN environment variable to the directory used for global usage.
2. Specify the **-g** flag along with the appropriate parameters to create a shortcut name definition.

   For example, to assign a shortcut name for system-wide usage to a machine name (server):

   ```
   1. set DFMADMN=D:\DFMADMN\GLOBAL
   2. dfmnet -g \\machine_name \\shortcut_name
   ```

   Alternatively, to assign a shortcut name for system-wide usage to a machine name **and** a prefix name for remote files:

   ```
   1. set DFMADMN=D:\DFMADMN\GLOBAL
   2. dfmnet -g \\machine_name\prefix_name \\shortcut_name
   ```

Displaying the Active Shortcut Names: By issuing the *dfmnet* command without any parameters, you can view all shortcut names that DFM recognizes as active. DFM locates these shortcut names in the active private and global *dfmnet* control files. The DFMADMN and DFMUSER environment variables, if defined, specify the directories for the global and private control files, respectively. If DFMUSER is undefined, the current directory is searched for a private control file.

Shortcut definitions from the private *dfmnet* control file override definitions from the global *dfmnet* control file.

Note: If the user wants to activate only shortcut names from the global *dfmnet* control file, the DFMUSER environment variable should be undefined and a private *dfmnet* control file (*dfmnet.dfm*) should not exist in the current directory.

The processing just described also occurs when DFM initializes its tables to access remote data.

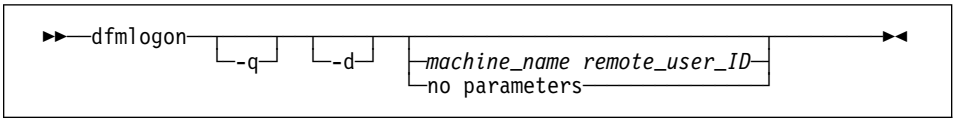
dfmlogon Command

The *dfmlogon* command provides DFM with a user ID and password to use when establishing a conversation with a particular server system. The command prompts you for a password, which is not displayed as you enter it. This information is accessible to all Windows applications that are activated from the user's workstation.

Note: **You must enter the *dfmlogon* command every time you log off or restart your Windows system.** When the next *dfmlogon* command is entered, the new information is available for use by all activated Windows applications.

You can repeat *dfmlogon* many times for the same server system. The new information replaces the existing information. The new logon information does not affect any conversations DFM has initiated. However, for any new conversation that DFM starts, (for example when DFM is called to process another VSAM API), DFM uses the new logon information.

Syntax



Flags

-q Use this flag if you do not want to display the title message.

-d Use this flag to delete remote logon information associated with the *machine_name*. *machine_name* is the only valid parameter with this flag.

Parameters

machine_name

The name of a server system. This name can be in LU alias format (1 to 8 characters) or in SNA network qualified address format (1 to 8 characters, a period, 1 to 8 characters). Eligible names are defined in the network software configuration files accessible to the user's workstation.

Note: The case of the machine name must match that of the target name specified in the DFM configuration. Likewise, it must also match the machine name in the application file name or shortcut name.

remote_user_ID

This is the user ID that *dfmlogon* will use when it accesses files on the server system.

This ID must be 1 to 8 characters.

no parameters

dfmlogon returns a list of all the machine names (server systems) associated with a user ID. *dfmlogon* does not display passwords.

Preparing to Run DFM-Based Applications (Some Reminders)

You must enter *dfmlogon* with the proper machine name, user ID, and password every time you log off or restart your Windows system.

The correct input of DFM setup information is crucial to running successful applications that call DFM for Windows. Before you start the actual application, it is recommended that you review the “active” DFM setup data. In the window where you plan to start the application:

1. Enter:

   ```
   echo %dfmuser%
   echo %dfmadmn%
   ```

   If you normally use one or both of these environment variables, do the responses display the directories you expect?

2. Enter:

```
dfmlogon
```

Do you see the correct machine name for the server system? Are the characters in the correct (upper or lower) case? Is the user ID spelled correctly?

3. Enter:

   ```
   dfmcfg -c | more
   ```

   You will see the contents of the active configuration file. Do you see the machine name for the server in the list of server systems? Are the MODE and SECURITY values for this machine name what you expect?

   If you want to convert the data in a remote file, do you see a FILE_DESCRIPTOR_MAP entry with a TARGET_FILENAME specifying this remote file name?

4. Enter:

   ```
   dfmnet
   ```

   If your application uses shortcut names for remote file names, does your shortcut name display the machine name for your server? Does the machine name match exactly (for example, upper or lower case) a machine name in the active configuration file (see step 3 above)?

If you are developing an application that uses DFM for Windows, detailed runtime error information is available through the VSAM API, DDMGetReplyMessage. Refer to *SMARTdata UTILITIES VSAM Application Programming Interface Reference* for information on the VSAM APIs.

Other Environment Variables Used by DFM

DFM requires the CDRASRV environment variable to be properly set. Otherwise, a translation error will occur when DFM is invoked. This environment variable should have been set up during the installation of the DD&C software. However, if you encounter a translation problem, you can verify that CDRASRV is defined by entering:

```
echo %cdrasrv%
```

If this variable is not defined, check with the installation person (or read the installation information) to determine what directory you need to “set” CDRASRV to.

If you want to use the Japanese (Kanji) version of DFM's error messages, you can set the DFMLANG environment variable as follows:

```
set DFMLANG=2985
```

This is necessary only if the system setting on your workstation is not already set to this value. (Unless DFM detects “2985” through DFMLANG or the system setting, DFM uses the English version of the messages.)

Chapter 4. Converting Record File Data

The incompatibility of data formats in different operating systems makes sharing data files impossible without data conversion. For example, data from a program written in COBOL or PL/1 on an MVS system is not readable by C language applications written on a Windows NT or OS/400 system. Data conversion enables heterogeneous systems to share record file data.

DFM for Windows provides data conversion that is transparent to application programs. It is implemented through an interface with Data Description and Conversion (DD&C) and A Data Language (ADL).

DD&C creates data descriptions of record file data, builds a conversion plan to map the data from one format to another, and executes the conversion plan when invoked by the source system.

ADL is the tool that DD&C uses to describe the data from the viewpoints of the different systems. ADL also creates the conversion plan that maps the conversion of data from one format to another.

DFM performs two types of conversion: character conversion of text records and data conversion of complex records.

In order to perform either type of data conversion for a remote file, the formatted configuration file used during DFM's initialization must have been created from a configuration text file that defined a FILE_DESCRIPTOR_MAP statement that corresponds to the remote file name. This statement gives the name of the conversion plan created by ADL for that remote file. Refer to Chapter 2, “The Configuration File” on page 11 for instructions on creating a text configuration file and FILE_DESCRIPTOR_MAP statements. Refer to “dfmcfg Command” on page 20 for instructions on creating the formatted configuration file that DFM uses during initialization.

Conversion of Text Data

Text data can be described as information represented as characters in a specific character set, such as ASCII or EBCDIC. Conversion of text data involves translating the characters from one character set to another.

To enable text conversion, the FILE_DESCRIPTOR_MAP statement must contain a BASE_CCSID definition that specifies the character set identifier (CCSID) describing the remote file data. The statement may also contain a VIEW_CCSID to specify the CCSID to which DFM will convert received records and from which DFM will convert sent records. If the statement does not provide a VIEW_CCSID, then DFM sets VIEW_CCSID to the CCSID corresponding to the active OEM code page on the client Windows system.

Conversion of Complex Data

Complex data can be described as information represented in multiple formats, possibly mixing text data with binary data. An example of complex data is a payroll file whose records consist of *text fields* containing name and address information, and *binary* fields containing numeric information such as salary and tax withholdings.

Overview of Complex Data Conversion

The end user invokes complex data conversion on a remote file DFM by providing a BASE_ADL_FILE and VIEW_ADL_FILE definition in the corresponding FILE_DESCRIPTOR_MAP statement.

- The BASE_ADL_FILE contains the name of the file giving the description of the remote data record (the base form).
- The VIEW_ADL_FILE contains the name of the file giving the description of the way the data should appear locally, after it is converted (the view form).

The statement may contain a VIEW_CPS_FILE definition to provide the name of the binary conversion space plan file DD&C for Windows will create from the base and view ADL files and will use during actual conversion. If there is no VIEW_CPS_FILE definition, DFM creates a file name based on the name of the view file.

The conversion is a two stage process. The first stage occurs when *dfmcfg* creates the formatted configuration file. For each file descriptor map definition in the input file, *dfmcfg* performs the following:

1. **Parse the View and Base ADL Files.**

   The *dfmcfg* command determines whether the ADL files have been updated since last parsed. If they have been updated, or if they have never been parsed, *dfmcfg* invokes DD&C for Windows to parse them.

   The *dfmcfg* command stores the binary files created by the parse in the same directory as the corresponding ADL file. The binary files have the same name as the ADL file with the *".adl"* extension replaced by either *".pln"* or *".dcl"*. If the ADL file does not end with *".adl"*, then *".pln"* or *".dcl"*. is appended to the ADL file name.

2. **Determine the Key Definitions.**

   If the file descriptor map refers to a keyed file, then *dfmcfg* invokes DD&C to build a declaration for the key field or fields for both the base and view ADL descriptions. This occurs only if the KEYLIST parameter is specified.

3. **Create the Conversion Plan Space File.**

   If no conversion plan space file exists or if the ADL files have changed since the conversion plan space file was last created, then *dfmcfg* invokes DD&C to create a new conversion plan space file.

   The *dfmcfg* command stores the binary conversion plan space file to the file name specified in VIEW_CPS_FILE in the FILE_DESCRIPTOR_MAP statement. See "Parameters Only for Complex Data Conversion" on page 16 for the case where

the FILE_DESCRIPTOR_MAP statement does not contain a VIEW_CPS_FILE parameter.

DD&C will use this conversion plan space file when performing the actual conversion of data records during remote file access.

The second stage of conversion occurs when an application uses DFM to open a remote file for complex data conversion. DFM invokes DD&C with the input or output record buffer. DD&C uses the conversion plan space created in the previous stage to convert the data from the view form to the base (or remote) form, as applicable.

Creating Base and View ADL Files

Use a text editor such as MS-DOS to create files containing ADL statements that describe the base and view record formats, respectively. See *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* for information on creating ADL data descriptions.

Both ADL files must have the basic structure:

```
DECLARE BEGIN ;
    record_name: SEQUENCE BEGIN ;
         field1:  <definition for field1> ;
         ...
         fieldn:  <definition for fieldn> ;
    END ;
END ;
```

The ADL files must adhere to the following restrictions:

- Each file should contain only one declaration.
- The name assigned to the record in the base and the view ADL files can be different.
- For each field in the view record ADL declaration, there must be a corresponding field in the base record ADL declaration with the same name.
- Corresponding fields must be convertible into each other's formats. For example, *field1* in the view record ADL cannot be a text field while *field1* in the base record ADL is a floating-point number.

During conversion of records received from the target, DD&C converts *field1* from the remote record into *field1* of the view format, *field2* into *field2*, and so on. During conversion of records sent to the target, the conversion is reversed.

Creating ADL Descriptions for Keyed Files

If the remote file is to be accessed as a keyed file, then the FILE_DESCRIPTOR_MAP definition must use the KEYLIST parameter to indicate which fields are the keys. The key names listed must each be a unique identifier within the ADL files. For example, if the name *field1* appears only once in both the Base and View ADL files, then *field1* would be a valid key name. However, in the following example, the name *field1* occurs twice in the ADL declaration:

```
DECLARE BEGIN ;
    Keyed_Rec: SEQUENCE BEGIN ;
         field1:   <field1 declaration>
         field2:   SEQUENCE BEGIN ;
                   field1: <field2.field1 declaration>
                   field2: <field2.field2 declaration>
         END ;
    END ;
 END ;
```

In this example, the KEYLIST parameter must refer to Keyed_Rec.field1 to refer to the first instance of *field1* or *field2.field1* to refer to the second instance of *field1*.

Note: In a situation like Keyed_Rec.field1, where the fully-qualified identifier contains the record name, the Base and the View record ADL declarations must use the same record name. In this case, it is "Keyed_Rec."

Using DFM's Default Conversion Plans

DFM uses a default mapping between input and output fields for complex data conversion. Each input field is mapped to a corresponding output field. If you want to do something else, you need to create a customized conversion plan. For information on a customized conversion plan, see Appendix B, "Creating a Customized Conversion Plan" on page 73.

Special Considerations for Remote File Data Conversion

Some things to be aware of when converting remote files are:

View and Base Files

When explicitly specifying the VIEW_CPS_FILE parameter, be sure not to specify the same name for two different file descriptor maps that do not require the same data conversion. Two file descriptor maps require the same data conversion if and only if they have the same BASE_ADL_FILE and VIEW_ADL_FILE parameters and either (1) both do not have a KEYLIST parameter or (2) both have the same KEYLIST parameter.

Similarly, if two files have the same VIEW_ADL_FILE name, but either (1) they have different BASE_ADL_FILE names or (2) they have different KEYLIST parameters, then you must override DFM's name generation and explicitly provide two unique VIEW_CPS_FILE names.

Creating Files

When creating a remote file that has an associated File Descriptor Map entry, provide the values for the local view of the data for the record length and key definition attributes. DFM uses the ADL or CCSIDs provided to calculate the appropriate record length and key definition values for the remote file.

When specifying a default record, provide the data as it would appear in the local system. DFM converts it to the target system format. DFM will

not be able to create the file if the file descriptor map specifies text conversion to or from a CCSID representing a mixed-byte character set.

Requesting File Attributes

When requesting file attributes for a remote file with an associated file descriptor map, DFM returns the correct values for the local view of the data. For example, the value of record length will be the length of records returned to the application, not the length of the data as it is stored on the target system. Key definitions will be as they appear in the view file, not as they appear in the data set on the target system.

DFM returns an error reply message if the target system returns a value for a requested attribute that does not match the value predicted by the ADL that describes the base file.

Creating Files With Fixed-Length Records

Do not specify the fixed-length record attribute when creating data sets with associated file descriptor maps if there is any possibility that the conversion may yield records of varying lengths. For example, when a set of records all of the same length are converted into a mixed-byte character set, the conversion can yield records of varying length. The server system returns a record length error when the application attempts to write records of varying lengths into a file with the fixed-length record attribute.

Using Keyed Fields and Varying Lengths

Keys must always start on the same offset in each record. Therefore, do not select a key field that occurs after a field that can vary in length. For example, say *field1* is a text field and *field2* is the key. If *field1* is being converted into a mixed-byte character set, then the start of *field2* can vary from record to record.

Chapter 5. DFM Error Messages

If a problem occurs that you cannot resolve (while using DFM), gather the following information necessary to evaluate the problem:

1. Error messages
2. Failing component
3. Configuration file parameters
4. ADL files being used
5. Applications in use.

Contact the IBM Support Center to find out if a circumvention or correction exists. If none exists, it may be necessary to open an authorized program analysis report. (APAR).

When you submit APAR information to the Support Center Representative describe the problem as precisely as possible. Include all of the above information and anything else that you consider pertinent.

DFM VSAM Support Messages

During remote file access, DFM returns no error messages. Instead, DFM communicates errors through DFM Reply Messages that you can retrieve with the VSAM API DDMGetReplyMessage. See Chapter 6, “Information for the Application Programmer” on page 55 for a discussion of the Reply Message and error processing.

Error Messages for DFM Commands

The following messages are issued by DFM commands. They are presented in this chapter according to the command type:

- Common messages
- DFMNET messages
- DFMLOGON messages
- DFMCFG messages
- Diagnostic messages

The following messages indicate:

- The **severity** of the message:
  - Information
  - Warning
  - Problem
  - Error
- The **text** of the message as it appears on your screen.

  Some messages include a “hint.” This is a step-by-step guidance for the most likely cause of the problem. The steps are chosen so they can be invoked from the same MS-DOS prompt window where the DFM commands are being entered.

- The appropriate user **response** to the message, if any.

  This is the user response for the most likely cause of the problem. This states **what** you need to do. You might choose to take action in the MS-DOS prompt window (using the hint exactly as stated), or perform other analogous steps on your workstation.

Common Messages

5648-2012-1 EHNX_COPYRIGHT

Severity: Information

Text:

```
Licensed Materials - Property of IBM
Distributed FileManager for Windows
5639-B92 (c) Copyright IBM Corp. 1997.  All Rights Reserved
Version 2
```

Response: None

5648-2012-2 EHNX_MSGTBLERR

Severity: Warning

Text: Unable to load DFM's message table.

Hint: Enter “DIR DFMMSG.DLL /S” to find the directory or folder where the DFM message file is located. Then enter “ECHO %PATH%” . Do you see the directory for DFMMSG.DLL listed? If not, add it by entering “SET PATH = *directory*;%PATH%” .

Response: Verify that the DFMMSG.DLL that contains DFM's Message table has been installed.

5648-2012-3 EHNX_DEFAULTMSG

Severity: Warning

Text: A message must be printed, but the message text is not available.

Hint: Enter “DIR DFMMSG.DLL /S” to find the directory or folder where the DFM message file is located. Then enter “ECHO %PATH%” . Do you see the directory for DFMMSG.DLL listed? If not, add it by entering “SET PATH = *directory*; %PATH%” .

Response: Verify that the DMMSG.DLL that contains the DFM's message table has been installed.

5648-2012-4 EHNX_DEFAULTWARN

Severity: Warning

Text: A minor error occurred during processing, but the message text is not available.

Hint: Enter “DIR DFMMSG.DLL /S” to find the directory or folder where the DFM message file is located. Then enter “ECHO %PATH%” . Do you see the directory for DFMMSG.DLL listed? If not, add it by entering “SET PATH = *directory*; %PATH%” .

Response: Verify that the DFMMSG.DLL that contains DFM's message table has been installed.

5648-2012-5 EHNX_TOOMANYPARMS

Severity: Warning

Text: Too many parameters entered.

Response: Verify the command line syntax.

5648-2012-6 EHNX_INVALIDFLAG

Severity: Warning

Text: Invalid flag.

Response: Verify the command line syntax.

5648-2012-7 EHNX_INVALIDFILENAME

Severity: Warning

Text: Invalid file name. The number of characters exceeds the system limit for file names (250 characters for Windows 95, 255 for Windows NT).

Response: Correct the name and reissue the command.

5648-2012-8 EHNX_DEFAULTSEV

Severity: Problem

Text: An error occurred during processing, but the DFM message text is not available.

Hint: Enter “DIR DFMMSG.DLL /S” to find the directory or folder where the DFM message file is located. Then enter “ECHO %PATH%” . Do you see the directory for DFMMSG.DLL listed? If not, add it by entering “SET PATH = directory;%PATH%” .

Response: Verify that the DFMMSG.DLL that contains DFM's message table has been installed.

5648-2012-9 EHNX_LOGICERROR

Severity: Problem

Text: A severe logic error occurred.

Response: Contact IBM Support Center.

5648-2012-10 EHNX_ALLOCERROR

Severity: Problem

Text: Error allocating memory.

Hint: Cancel applications (or tasks) you do not need to have active. Search HELP in your system for troubleshooting - “if you run out of memory.”

Response: Reduce system requirements and retry command.

5648-2012-11 EHNX_BUSY

Severity: Problem.

Text: This command is being processed in another window and must complete.

Hint: Wait until that command in the other window completes. Re-issue the command you just entered.

Response: Follow the actions described above.

5648-2012-12 EHNX_SYSCALLERR

Severity: Problem

Text: An error occurred while invoking a system call.

Response: Contact IBM Support Center.

5648-2012-13 EHNX_OPENERR

Severity: Problem

Text: An error occurred while opening file (*filename*).

Hint: Enter “DIR *filename* /S” to find the directory for the file. If no file is found, check the spelling on the original DFM command that surfaced the error. If the file is found, verify that you have the correct access authority for the file.

Response: Verify that the file identified in the message exists and that you have the proper access authority.

5648-2012-14 EHNX_WRITEERR

Severity: Problem

Text: An error occurred while writing to file (*filename*). Verify that you have the authority to write to the file.

Response: Verify the authority as described above.

5648-2012-15 EHNX_READERR

Severity: Problem

Text: An error occurred while reading file (*filename*). Verify that you have the authority to read the file.

Response: Verify the authority as described above.

5648-2012-16 EHNX_CLOSEERR

Severity: Problem

Text: An error occurred while closing file (*filename*).

Hint: Enter “DIR *filename* /S” to find the directory for the file. If the file is found, verify that you have write access to the file.

Response: Verify that you have the authority to write to the file.

5648-2012-17 EHNX_DEFAULTSYS

Severity: Problem

Text: A system error occurred during processing, but the DFM message text is not available.

Hint: Enter “DIR DFMMSG.DLL /S” to find the directory or folder where the DFM message file is located. Then enter “ECHO %PATH%” . Do you see the directory for DFMMSG.DLL listed? If not, add it by entering “SET PATH = directory;%PATH%” .

Response: Verify that the DLL that contains DFM's message table has been installed.

5648-2012-18 EHNX_DELFAIL

Severity: Error

Text: Unable to delete file *filename*. Verify that you have the correct permission to delete the file.

Response: Verify as described above.

5648-2012-48 EHNX_EVNERRLOC

Severity: Error

Text: The DFMUSER environment variable is undefined.

Hint: Enter “SET DFMUSER=*directory*” where *directory* is the directory available for private usage.

Response: Set DFMUSER environment variable to the private directory and reissue the command.

5648-2012-49 EHNX_EVNERRGLB

Severity: Error

Text: The DFMADMN environment variable is undefined.

Hint: “SET DFMADMN=*directory*” where *directory* is the directory available for global usage.

Response: Set DFMADMN environment variable to the global directory and reissue the command.

5648-2012-50 EHNX_NEPRVDIR

Severity: Error

Text: The private directory defined by the DFMUSER environment variable does not exist.

Hint: Enter “ECHO %DFMUSER%” . Check the spelling of the directory returned in this response. Enter “SET DFMUSER=*directory*” to make a correction.

Response: Set DFMUSER environment variable to an existing directory for private usage.

5648-2012-51 EHNX_NEGLBDIR

Severity: Error

Text: The global directory defined by the DFMADMN environment variable does not exist.

Hint: Enter “ECHO %DFMUSER%” . Check the spelling of the directory returned in this response. Enter “SET DFMUSER=*directory*” to make a correction.

Response: Set DFMUSER environment variable to an existing directory for private usage.

5648-2012-19 •5648-2012-25

DFMNET Messages

5648-2012-19 DFMNET_USAGE

Text: Syntax: dfmnet [-d] [-g] [-q] [\\machine_name [\path_name]]\\shortcut_name]

See the SdU Windows User's Guide, Order No. SC26-7134, for more information.

Response: Reissue *dfmnet* with the correct syntax.

5648-2012-20 DFMNET_ALREADYASSIGNED

Severity: Information

Text: (*shortcut_name*) is already assigned.

Hint: Enter "*dfmnet*" to see current DFM shortcut name assignments. Either reissue *dfmnet* with a different shortcut name or delete the original shortcut name assignment. Enter "DFMNET -D *shortcut_name*" to delete the shortcut name you want to change. Then enter the original *dfmnet* command again to re-define the name.

Response: Either create a new shortcut name or issue *dfmnet -d shortcut_name*, then reissue *dfmnet.*

5648-2012-21 DFMNET_ASSIGNOK

Severity: Information

Text: *dfmnet* assignment of (*shortcut_name*) to (*machine_name**pathname or* *machine_name*) completed successfully.

Response: None.

5648-2012-22 DFMNET_DISPLAYLOC

Severity: Information

Text: Private list of shortcut names:

Response: None.

5648-2012-23 DFMNET_DISPLAYGLOB

Severity: Information

Text: Global list of shortcut names:

Response: None.

5648-2012-24 DFMNET_NOASSIGN

Severity: Information

Text: None defined. (No *shortcut* names are defined.)

Response: None.

5648-2012-25 DFMNET_ISASSIGNED

Severity: Information

Text: (*shortcut_name*) - (*machine_name**prefix_name or* *machine_name*)

Response: None.

5648-2012-26 DFMNET_BADSHORTCUTNAME_DBLSLASHES

Severity: Warning

Text: (*shortcut_name*) is an invalid shortcut name. The name must be preceded by double back-slashes (\\).

Response: Reissue *dfmnet* with the correct shortcut name.

5648-2012-27 DFMNET_BADSHORTCUTNAME_LENGTH

Severity: Warning

Text: (*shortcut_name*) is an invalid shortcut name. The name must consist of 1-8 characters.

Response: Reissue *dfmnet* with the correct shortcut name.

5648-2012-28 DFMNET_NOTUNCFMT_LENGTH

Severity: Warning

Text: A qualifier in the prefix name is invalid. Each qualifier must consist of 1-8 characters.

Response: Correct the prefix name qualifier, then reissue *dfmnet*.

5648-2012-29 DFMNET_BADSHORTCUTNAME_CHARSET

Severity: Warning

Text: (*shortcut_name*) is an invalid shortcut name. The name contains at least one character that is not in the following character set (inside the braces):

{Character_Set_For_Shortcut_Name}

Response: Reissue *dfmnet* with the correct shortcut name.

5648-2012-30 DFMNET_SNA2DOTS

Severity: Warning

Text: Invalid SNA network address. The address must contain exactly one decimal point.

Response: Correct the SNA network address, then reissue *dfmnet*.

5648-2012-31 DFMNET_LULENBAD

Severity: Warning

Text: Invalid SNA Logical Unit alias. A SNA LU alias must consist of 1-8 characters.

Response: Correct the SNA LU alias, then reissue *dfmnet*.

5648-2012-32 DFMNET_SNAADDRBAD_LENGTH

Severity: Warning

Text: Invalid SNA network address format. The address must consist of 3-17 characters.

Response: Correct the SNA network address format, then reissue *dfmnet*.

5648-2012-33 DFMNET_NOTUNCFMT_DBLSLASHES

Severity: Warning

Text: A parameter is invalid. The parameter must be preceded by double backslashes (\\).

Response: Reissue *dfmnet* with the correct parameter.

5648-2012-34 DFMNET_NOTUNCFMT_CHARSET

Severity: Warning

Text: The prefix name is invalid. The name contains at least one character that is not in the following character set (inside the braces):

{Character_Set_For_Prefix_Pathname}

Response: Reissue *dfmnet* with the correct prefix name.

5648-2012-35 DFMNET_NOTASSIGNEDLOC

Severity: Warning

Text: *shortcut_name* is not in the private *dfmnet* control file.

Response: None.

5648-2012-36 DFMNET_NOTASSIGNEDGLOB

Severity: Warning

Text: *shortcut_name* is not in the global *dfmnet* control file.

Response: None.

5648-2012-37 DFMNET_NOTUNCFMT_PATHLENGTH

Severity: Warning

Text: The prefix name is invalid. The number of characters exceeds the system limit for file names (250 characters for Windows 95, 255 for Windows NT).

Response: Correct the prefix name, then reissue *dfmnet*.

5648-2012-38 DFMNET_REMOVED

Severity: Information

Text: *dfmnet* successfully removed *shortcut_name*.

Response: None.

5648-2012-40 DFMNET_OPENFAIL

Severity: Problem

Text: Unable to open *dfmnet* control file.

Hint: Enter “DIR DFMNET.DFM /S” to find directories or folders containing this filename. (1) If you specified “-g” on the *dfmnet* command, then you want to access the global *dfmnet* control file. Enter “ECHO %DFMADMN%” . Does the reply show the directory for the proper global DFMNET.DFM? If not, enter “SET DFMADMN=*directory*” to define the proper directory. (2) If you want to use a private DFMNET.DFM file (no “-g” flag), select the proper directory for this file and

enter “SET DFMUSER= *directory*” . (3) In either case, if the command still fails, check that you have the correct access permission for the file.

Response: Verify that the file exists and that you have authority to read or write to it.

5648-2012-41 DFMNET_ALLOCFAIL

Severity: Problem

Text: An error occurred while allocating memory.

Hint: Cancel applications (or tasks) you do not need to have active. Search HELP in your system for troubleshooting - “if you run out of memory.”

Response: Reduce memory requirements and reissue the *dfmnet* command.

5648-2012-42 DFMNET_READFAIL

Severity: Problem

Text: An error occurred while reading the *dfmnet* control file. Verify that you have authority to read the file.

Response: Verify as described above.

5648-2012-43 DFMNET_INCORFMT

Severity: Problem

Text: The *dfmnet* control file format is incorrect.

Hint: The *dfmnet* control file in error maybe either or both the private DFMNET.DFM or the global DFMNET.DFM file. Here is a method to determine which file is in error. Enter “ECHO %DFMUSER%” to see the directory for the private DFMNET.DFM file. If the DFMUSER environment variable is undefined, look for DFMNET.DFM in your current directory. Then temporarily rename this file. For example, enter “RENAME *directory*\DFMNET.DFM *directory*\DFMNET.DF1” . Then repeat the *dfmnet* command. If the command returns normally, then you know this file is bad. If the same error occurs, the bad file must be the global DFMNET.DFM. To confirm, enter “ECHO %DFMADMN%” to get the directory for the global DFMNET.DFM file. Enter “SET DFMADMN=” and repeat the *dfmnet* command. The bad file will have to be re-created using *dfmnet* to re-define each valid shortcut name in the file.

Response: Rename or delete the file and reissue the *dfmnet* command.

5648-2012-44 DFMNET_OPENWRITEFAIL

Severity: Problem

Text: Unable to open the *dfmnet* control file for update.

Hint: Enter “DIR DFMNET.DFM /S” to find directories or folders containing this filename. (1) If you specified “-g” on the *dfmnet* command, then you want to access the global *dfmnet* control file. Enter “ECHO %DFMADMN%” . Does the reply show the directory for the proper global DFMNET.DFM? If not, enter “SET DFMADMN=*directory*” to define the proper directory. (2) If you want to use a private DFMNET.DFM file (no “-g” flag), select the proper directory for this file and enter “SET DFMUSER= *directory*” . (3) Re-enter the original *dfmnet* command. If it still fails, check that you have write access to the file.

Response: Verify that the file exists and that you have write access.

5648-2012-45 DFMNET_WRITEFAIL

Severity: Problem

Text: Unable to write to *dfmnet* control file. Verify that you have write access to the file.

Response: Perform the corrective action described by the message text.

5648-2012-46 DFMNET_CLOSEFAIL

Severity: Problem

Text: An error occurred while closing the *dfmnet* control file.

Hint: (1) If you specified “-g” on the *dfmnet* command, enter “ECHO %DFMADMN%” . (2) If you specified a private DFMNET.DFM file (no “-g” flag), enter “ECHO %DFMUSER%” . (3) If you see a directory in response to this command, do you have write access to this directory, and to the DFMNET.DFM file in it? (4) If the ECHO command does not return a directory, check that you have write access to the current directory.

Response: Verify that you have write access to the file.

5648-2012-47 DFMNET_SNAADDRBAD_CHARSET

Severity: Warning

Text: Invalid machine name. The machine name contains at least one character that is not in the following character set (inside the braces):

{Character_Set_For_SNA_Network_Address}

Response: Reissue the *dfmnet* command with the correct SNA network address.

DFMLOGON Messages

5648-2012-54 DFMLOGON_USAGE

Severity: Information

Text: Syntax: dfmlogon [-q] [-d] [machine_name user_ID]

See the SdU Windows User's Guide, Order No. SC26-7134, for more information.

Response: Follow the directions noted above.

5648-2012-55 DFMLOGON_DISPHEAD

Severity: Information

Text: List of valid DFM logon control table entries:

Response: None.

5648-2012-56 DFMLOGON_DISPLINE

Severity: Information

Text: *(machine_name) - (user_ID)*

Response: None.

5648-2012-57 DFMLOGON_NOENTRIES

Severity: Information

Text: No valid entries found.

Response: None.

5648-2012-58 DFMLOGON_DELSUCCESS

Severity: Information

Text: Successfully deleted logon control table entry for (*machine_name*).

Response: None

5648-2012-59 DFMLOGON_ADDSUCCESS

Severity: Information

Text: Successfully added/modified logon control table entry for *(machine_name).*

Response: None

5648-2012-60 DFMLOGON_PWDPROMPT

Severity: Action

Text: “Enter password” .

Response: Enter password at the prompt. Your password will not be displayed as you type it.

5648-2012-61 DFMLOGON_BADUID

Severity: Warning

Text: The user ID parameter is too long. The ID must consist of 1-8 characters.

Response: Correct the userid parameter.

5648-2012-62 DFMLOGON_BADPWD

Severity: Warning

Text: The password is too long. The password must consist of 1-8 characters.

Response: Change the password.

5648-2012-64 DFMLOGON_DISPBADUID

Severity: Warning

Text: Warning: User ID for above entry is invalid or missing.

Response: Reissue *dfmlogon* for the server system to reenter the logon information.

5648-2012-65 DFMLOGON_DISPBADPWD

Severity: Warning

Text: Warning: Password for above entry is missing.

Response: If a password is required for this server system, then issue *dfmlogon* to reenter the logon information for the server system.

5648-2012-66 DFMLOGON_NOTFOUND

Severity: Warning

Text: Unable to find DFM machine name in logon control table.

Hint: Enter *dfmlogon* to view current DFM machine names in this table. Note: machine names are case-sensitive. Re-enter the original *dfmlogon* command exactly as it is spelled in the table.

Response: Enter *dfmlogon* to provide logon information for the server system.

5648-2012-76 DFMLOGON_ALLOCFAIL

Severity: Problem

Text: Could not allocate memory to process logon control table.

Hint: cancel applications (or tasks) you do not need to have active. Search HELP in your system for troubleshooting - “if you run out of memory.”

Response: Reduce system resources and retry the command.

5648-2012-77 DFMLOGON_STARTPROCESS_FAILED

Severity: Problem

Text: Unable to start the logon control process.

Hint: enter “DIR DFM*.DLL /S” to find the directory or folder for DLLs which are part of DFM, and “DIR DFM*.EXE /S” for the directory for EXEs which are part of DFM. Enter “ECHO %PATH%” . Are these directories listed? If not enter “SET PATH = *directory*;%PATH%” to add them.

Response: Ensure that the directory containing DFM's *.exe and *.dll files are listed in the PATH environment variable.

5648-2012-78 DFMLOGON_CREATEDFMLOGONFILEFAILED

Severity: Problem

Text: Insufficient space is available on the hard drive.

Hint: Enter “ECHO %DFMUSER%” . Is a directory defined? If so, enter “DIR *directory*” to see how many unused bytes are available. If no directory is defined, enter “DIR” to see the number of unused bytes in the current directory. Delete unnecessary files from the drive associated with this directory until a “DIR *directory*” indicates there are more than 4096 unused bytes.

Response: Free space on the hard drive and retry the command.

5648-2012-79 DFMLOGON_TMPOPENFAIL

Severity: Problem

Text: Insufficient space is available on the hard disk.

Hint: Enter “ECHO %DFMUSER%” . Is a directory defined? If so, enter “DIR *directory*” to see how many unused bytes are available. If no directory is defined, enter “DIR” to see the number of unused bytes in the current directory. Delete unnecessary files from the drive associated with this directory until a “DIR *directory*” indicates there are more that 4096 unused bytes.

Response: Free space on the hard drive and retry the command.

5648-2012-131 DFMLOGON_MACHCASE1

Severity: Information

Text: The machine name was found in the active DFM configuration file; it is NOT associated with a shortcut name.

Response: None.

5648-2012-132 DFMLOGON_MACHCASE2

Severity: Information

Text: The machine name was found in the active DFM configuration file and it is associated with at least one shortcut name.

Response: None.

5648-2012-133 DFMLOGON_MACHCASE3

Severity: Warning

Text: The machine name was NOT found in the active DFM configuration file, but it is associated with at least one shortcut name.

Hint: Enter "dfmcfg -c" to see the contents of the active DFM configuration file.

Response: The default conversational settings displayed by issuing the "dfmcfg -c" command are assigned to the specified machine name. Verify that the default conversational settings are acceptable before using DFM to access the server system.

5648-2012-134 DFMLOGON_MACHCASE4

Severity: Warning

Text: The machine name was NOT found in the active DFM configuration file, and it is NOT associated with any shortcut name.

Hint: Enter "dfmcfg -c" to see the contents of the active DFM configuration file. Enter "dfmnet" to see the list of active shortcut names and their associated machine names.

Response: The default conversational settings displayed by issuing the "dfmcfg -c" command are assigned to the specified machine name. Verify that the default conversational settings are acceptable before using DFM to access the server system.

DFMCFG Messages

5648-2012-89 DFMCFG_USAGE

Severity: Information

Text: Syntax: dfmcfg [-c] [-i] [-q] source_file [output_file]

See the SdU Windows User's Guide, Order No. SC26-7134, for more information.

Response: Follow the directions in the message.

5648-2012-90 DFMCFG_SUCCESS

Severity: Information

Text: Successfully created configuration file *file_name*.

Response: None.

5648-2012-91 DFMCFG_DDCPARSEOK

Severity: Information

Text: DD&C parse successful for file *file_name*.

Response: None.

5648-2012-92 DFMCFG_DDCCPBOK

Severity: Information

Text: DD&C successfully created conversion plan space file *file_name*.

Response: None.

5648-2012-93 DFMCFG_DDCWARN

Severity: Warning

Text: Error encountered by DD&C. Conversion plan space was not created. Processing continues.

Response: Correct DD&C error and reissue the *dfmcfg* command.

5648-2012-94 DFMCFG_EMPTYINPUT

Severity: Warning

Text: Warning: Input file contains no statements.

Hint: Using the input file name of the *dfmcfg* command, view this file. It should contain text with keywords like DFM_TARGET and/or FILE_DESCRIPTOR_MAP, and all comments should begin with "/*" and end with "*/." See the SdU Windows User's Guide, Order No. SC26-7134, for more information.

Response: Verify the contents of the input file.

5648-2012-95 DFMCFG_PARSEERROR

Severity: Error

Text: Error parsing configuration input file.

Hint: Look at the file name returned from this command and the two values in parentheses. The first value is the line number that contains an error, and the second value indicates the column position where the error was detected. Correct the error and try the *dfmcfg* command again. See the SdU Windows User's Guide, Order No. SC26-7134, for more information.

Response: Correct syntax error and reissue the *dfmcfg* command.

5648-2012-97 DFMCFG_BADMSL

Severity: Error

Text: Invalid value for maximum send limit. This value must be an integer between 256 and 32767.

Response: Correct the value and reissue the *dfmcfg* command.

5648-2012-98 DFMCFG_BADMODENAME

Severity: Error

Text: Invalid value for mode name. The name must consist of 1-8 characters.

Response: Correct the value and reissue the *dfmcfg* command.

5648-2012-99 DFMCFG_DDCPARSEERROR

Severity: Error

Text: DD&C Parser returned an error for file *file_name*. Refer to DD&C list file *file_name* for more information. The DD&C Condition Token has the following contents:

Message Severity:

Message Number:

Service Condition Case:

Condition Severity:

Control:

Facility ID:

Instance Specific:

Response: See *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* to correct the ADL file.

5648-2012-100 DFMCFG_DDCCPBERROR

Severity: Error

Text: DD&C Conversion Plan Builder returned an error. The DD&C Condition Token has the following contents:

Message Severity:

Message Number:

Service Condition Case:

Condition Severity:

Control:

Facility ID:

ADL Communication Area:

Length:

Exception Identifier:

Severity Code:

Plan Identifier:

Plan Statement:

Input Error Date:

Source Field Identifier:

Target Field Identifier:

Response: See *SMARTdata UTILITIES Data Description and Conversion* for information on the Conversion Plan Builder API.

5648-2012-101 DFMCFG_DDCERR

Severity: Error

Text: Terminal error encountered by DD&C. Processing is halted.

Response: Contact the IBM Support Center.

5648-2012-102 DFMCFG_DDCKEYERR

Severity: Error

Text: A filed listed as a key was not found in a declare space.

Response: Verify that all identifiers listed in the KEYLIST of the FILE_DESCRIPTOR_MAP correspond to a uniquely identified qualifier in both the view and the base ADL files.

5648-2012-103 DFMCFG_CDRAERR

Severity: Error

Text: Error encountered by DD&C character conversion routine (CDRA).

Response: Contact the IBM Support Center.

5648-2012-104 DFMCFG_CCSIDNOTAVAIL

Severity: Error

Text: CCSID <*number*> is not available in the DD&C character conversion routine (CDRA) resource repository.

Response: Verify that the CCSID value is valid. Try to substitute an equivalent CCSID. See the *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* for a list of valid CCSID values.

5648-2012-105 DFMCFG_CCSIDINVALID

Severity: Error

Text: CCSID <*number*> is outside of the permitted range.

Response: Verify that the CCSID value is valid. See the *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* for a list of valid CCSID values.

5648-2012-106 DFMCFG_CDRAFAILED

Severity: Error

Text: The DD&C character conversion routine (CDRA) failed with the following return codes: Status Code <*number*> Reason Code <*number*>

Response: See the *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* for corrective action.

5648-2012-107 DFMCFG_DDCUTILFAIL

Severity: Error

Text: A DD&C utility function failed with the following return codes: Message Severity <*number*> Message Number <*number*>

Response: Contact the IBM Support Center.

5648-2012-108 DFMCFG_DCLSPCBAD

Severity: Error

Text: A data conversion space is invalid.

Response: The succeeding message will provide the current line in the configuration file being processed. Delete all files with the extension .dcl that corresponds to the ADL files in the FILE_DESCRIPTOR_MAP being processed. Rerun the *dfmcfg* command.

5648-2012-109 DFMCFG_NOKEYWORD

Severity: Error

Text: A required keyword is missing.

Response: Verify that the configuration file follows the correct syntax.

5648-2012-110 DFMCFG_OPENCOMMENT

Severity: Error

Text: A comment does not close before the end of the file.

Response: Verify that all comments are closed with a */.

5648-2012-111 DFMCFG_GETCURRENTDIR_FAILED

Severity: Error

Text: Unable to determine the current directory.

Response: Contact the IBM Support Center.

5648-2012-112 DFMCFG_INVALID_TSLE

Severity: Error

Text: Invalid DFM_TARGET entry detected in the formatted configuration file (*dfmcfg.dfm*).

Response: Issue the dfmcfg command to recreate the formatted configuration file. Contact the IBM Support Center if the problem persists.

5648-2012-113 DFMCFG_INVALID_FMLE

Severity: Error

Text: Invalid FILE_DESCRIPTOR_MAP entry detected in the formatted configuration file (*dfmcfg.dfm*).

Response: Issue the dfmcfg command to recreate the formatted configuration file. Contact the IBM Support Center if the problem persists.

5648-2012-114 DFMCFG_INVALID_SACB

Severity: Error

Text: Invalid DFM system anchor control block detected in the formatted configuration file (*dfmcfg.dfm*).

Response: Issue the dfmcfg command to recreate the formatted file. Contact the IBM Support Center if the problem persists.

5648-2012-115 DFMCFG_DEFDCLSPEC_NOTFOUND

Severity: Error

Text: The default declared space file (dfmdclsp.dcl) was not found.

Response: Verify that the default declared space file exists in the directory where the SdU for Windows DLLs and EXEs are located. Verify that this directory is covered by the PATH environment variable. Contact the IBM Support Center if the default declared space file was not included in the installation package.

5648-2012-116 DFMCFG_DEFPLNSPEC_NOTFOUND

Severity: Error

Text: The default plan space file (dfmplnsp.pln) was not found.

Response: Verify that the default plan space file exists in the directory where the SdU for Windows DLLs and EXEs are located. Verify that this directory is covered by the PATH environment variable. Contact the IBM Support Center if the default plan space file was not included in the installation package.

5648-2012-117 DFMCFG_DEFKEYPLNSPC_NOTFOUND

Severity: Error

Text: The default key plan space file (dfmkeysp.pln) was not found.

Response: Verify that the default key plan space file exists in the directory where the SdU for Windows DLLs and EXEs are located. Verify that this directory is covered by the PATH environment variable. Contact the IBM Support Center if the default key plan space file was not included in the installation package.

5648-2012-120 •5648-2012-126

Diagnostic Messages

5648-2012-120 DIAG_LABEL_DESCRIPTION

Severity: Information

Text: Problem Description:

Response: None

5648-2012-121 DIAG_DFM_FAILED

Severity: Information

Text: DFM Routine: (Name_of_failing_DFM_Routine)

Response: None

5648-2012-122 DIAG_API_FAILED

Severity: Information

Text: Failing API: (Name_of_failing_API)

Response: None

5648-2012-123 DIAG_GETLASTERROR

Severity: Information

Text: GetLastError: (Return_Code_From_Windows_GetLastError_API)

Response: None

5648-2012-124 DIAG_STATE

Severity: Error

Text: Unexpected error detected while validating the command line.

Response: Contact the IBM Support Center.

5648-2012-125 DIAG_SYSTEM_INTEGRITY

Severity: Error

Text: Invalid default security value detected in the DFM System control block.

Response: Issue the dfmcfg command to recreate the formatted configuration file. Contact the IBM Support Center if the problem persists.

5648-2012-126 DIAG_KEYWORD_TABLE

Severity: Error

Text: Valid keyword is not defined in DFM keyword table.

Response: Contact the IBM Support Center.

5648-2012-127 DIAG_SERVER_SECURITY

Severity: Error

Text: Invalid security value detected in DFM target control block.

Response: Issue the dfmcfg command to recreate the formatted configuration file. Contact the IBM Support Center if the problem persists.

5648-2012-128 DIAG_UNKNOWN_TASK

Severity: Error

Text: End user command cannot service request since the task to perform is unknown.

Response: Contact the IBM Support Center.

5648-2012-129 DIAG_FDM_DELETE

Severity: Error

Text: File Descriptor Map entry to delete could not be found.

Response: Contact the IBM Support Center.

5648-2012-130 DIAG_KEYLIST_ENTRY

Severity: Error

Text: An invalid Key List entry was detected.

Response: Contact the IBM Support Center.

Chapter 6. Information for the Application Programmer

This section lists the VSAM APIs supported by DFM for Windows and provides information about DFM Reply Messages and error processing.

VSAM API commands

The following list shows all the VSAM APIs supported by DFM for Windows. These APIs are described in *SMARTdata UTILITIES VSAM Application Programming Interface Reference*. Also in the VSAM reference is the information returned to the caller of the API for error conditions.

Function Call	Description
DDMClose	Close File
DDMCreateAltIndex	Create Alternate Index File
DDMCreateRecFile	Create Record File
DDMDelete	Delete File
DDMDeleteRec	Delete Record
DDMForceBuffer	Commit a File's Cached Information
DDMGetRec	Retrieve a Record
DDMGetReplyMessage	Returns reply messages for prior DDM calls
DDMInsertRecEOF	Insert a Record at End of File
DDMInsertRecKey	Insert a Record by Key Value
DDMInsertRecNum	Insert a Record by Record Number
DDMLoadFileFirst	Load First Record into a File
DDMLoadFileNext	Load Next Record into a File
DDMModifyRec	Modify Record
DDMOpen	Open a File
DDMQueryFileInfo	Retrieve Information about a File
DDMQueryPathInfo	Retrieve Information about a File or Directory
DDMRename	Rename a File
DDMSetBOF	Set Cursor to Beginning of File
DDMSetEOF	Set Cursor to End of File
DDMSetFileInfo	Set Information about a File
DDMSetFirst	Set Cursor to First Record in File
DDMSetKey	Set Cursor by Key

DDMSetKeyFirst	Set Cursor to First Record in Key Sequence
DDMSetKeyLimits	Set Key Limits
DDMSetKeyLast	Set Cursor to Last Record in Key Sequence
DDMSetKeyNext	Set Cursor to Next Record in Key Sequence
DDMSetKeyPrevious	Set Cursor to Previous Record in Key Sequence
DDMSetLast	Set Cursor to Last Record in File
DDMSetMinus	Set Cursor Minus
DDMSetNextKeyEqual	Set Cursor to Next Record with Equal Key
DDMSetNextRec	Set Cursor to Next Record
DDMSetPathInfo	Set a File's or a Directory's Information
DDMSetPlus	Set Cursor Plus
DDMSetPreviousRec	Set Cursor to Previous Record
DDMSetNextKeyEqual	Set Cursor to Next Record with Equal Key
DDMSetRecNum	Set Cursor to Record Number
DDMSetUpdateKey	Set Update Intent by Key
DDMSetUpdateNum	Set Update Intent by Record Number
DDMUnloadFileFirst	Unload First Record from File
DDMUnloadFileNext	Unload Next Record from File
DDMUnLockRec	Unlock Implicit Record Lock

DFM Reply Messages and Error Processing

When DFM receives notice of an error condition by the server system in response to a file access request, it does not issue an error message. However, it does return an error code to the caller and can make the DFM Reply Message structure accessible. Within this structure, the type of error is encoded in a two-byte hexadecimal value called a code point. Depending on the specific reply message and the server implementation, more “server diagnostic” information in addition to the error codepoint can be returned. Table 3 on page 59 lists the DFM Reply Message names alphabetically, its hexadecimal code point, decimal code point equivalents, and a short description of the DFM access request feedback.

The choice of programming language used by the application programmer dictates the amount of detailed knowledge of Reply Messages required for error processing when accessing files on the server system. Likewise, the choice of programming language dictates the amount of detailed Reply Message information available.

The C Programmer

The C programmer can use the VSAM API DDMGetReplyMessage following a VSAM API call to solicit the specific error Reply Message. *SMARTdata UTILITIES VSAM Application Programming Interface Reference* documents this interface and the Reply Message structure as well as a detailed discussion of many, but not all, Reply Messages.

The COBOL Programmer

The COBOL programmer can code programs based on the standardized COBOL returned file status values. If the COBOL runtime library invokes the VSAM APIs on behalf of the application program to do remote file access, it must map the Reply Message meaning to the most appropriate standardized COBOL file status. The IBM VisualAge COBOL product for the workstation provides for a second file status field which the COBOL programmer can optionally choose to define to provide program access to the specific DFM error. This is simply done by defining the second file status with a minimum length of six bytes. The fifth and sixth bytes in the field contains a binary decimal representation of the Reply Message codepoint. (The first four bytes are a decimal number indicating the total length of the Reply Message structure.)

The following examples from a COBOL program:

- Associates the second file status FILE8-STATUS with FILE1
- Shows a sample definition for FILE8-STATUS which contains the Reply Message
- Illustrates a DISPLAY of the COBOL file status and DFM Reply Message values.

For simplicity purposes, the DFM Reply Message displays as a decimal number. Note, before running the program, the environment variable DDDIRECT is set to assign a specific file name, for example:

```
SET DDDIRECT=\\MVS1\USER1.DDM.KEYFILE
```

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT FILE1 ASSIGN TO DDDIRECT
   ORGANIZATION IS INDEXED
   ACCESS MODE IS DYNAMIC
   RECORD KEY IS EMP-NAME
   FILE STATUS IS FILE1-STATUS FILE8-STATUS.
.
.
WORKING STORAGE SECTION.
01  FILE1-STATUS.
    05  FILE1-STATUS-A    PIC X.
    05  FILE1-STATUS-B    PIC X.
01  FILE8-STATUS.
    05  FILE8-STATUS-LEN  COMP PIC S9(8).
    05  FILE8-STATUS-CP   COMP PIC S9(4).
.
.
PROCEDURE DIVISION.
OPEN INPUT FILE1.
IF FILE1-STATUS NOT EQUAL TO GOOD-STATUS THEN
   DISPLAY "Error opening INPUT file.  Status code = ",
           FILE1-STATUS, " Dataname-8 = " FILE8-STATUS-CP
.
```

Because the file does not exist on the server, the “OPEN INPUT FILE1” fails: The following is the output from such a run:

```
Error opening INPUT file. Status code = 35 Dataname-8 = 4622
```

Table 3 on page 59 lists DFM Reply Message names, hexadecimal code points, decimal code point equivalents, and a short description of the DFM error. The COBOL file status value contained in FILE1-STATUS is 35 which means “An OPEN statement with the INPUT, I-O, or EXTEND phrase was attempted on a non-optional file that was not present.” The decimal equivalent of the DFM Reply Message codepoint value contained in the FILE8-STATUS-CP is 4622. This decimal value can be found in Table 3 on page 59 in the row for Reply Message FILNFNRM with hexadecimal codepoint X'120E' and meaning “File Not Found.”

The PL/I Programmer

The PL/I programmer sets the file name environment variable “dd:” followed by the generic file name coded in the program and its specific real file name. To trigger DFM remote access of files, the access method specification of *amthd(remote)* must be set with the file name. An example of setting the environment variable is as follows, the key being the AMTHD keyword.

```
set dd:infile=\\mvs2\edwards.io600.dat,amthd(remote)
```

In the above statement, the file name in the PL/I program is *infile* and the real file name is **edwards.io600.dat**. The real file is on an MVS system called **mvs2**. To continue this example, suppose the user ID for whom the remote file access is being attempted

is RACF prohibited from accessing the file. The server system will return an error when the OPEN request is made to indicate it could not OPEN the file on behalf of the user. PL/I reports the PL/I return code (Subcode1) and the hexadecimal DFM Reply Message value (Subcode2) in an error message. The following is the output from such a run:

```
IBM0265I  ONCODE=0099  The UNDEFINEDFILE condition was raised
          because the file could not be opened
          Subcode1=00024  Subcode2=123B  ( FILE= or ONFILE= F).
   At offset +00000145 in procedure with entry CIO627B
```

Table 3 lists DFM Reply Message names, hexadecimal code points, decimal code point equivalents, and a short description of the error. The hexadecimal *Subcode2* value in the example above of 123B corresponds to the row for Reply Message FILATHRM, decimal value 4667, and meaning “Not Authorized to File.”

The commonly encountered DFM Reply Messages are listed alphabetically in Table 3.

Table 3 (Page 1 of 3). DFM Reply Messages

Reply Message ID	Hexadecimal	Decimal	Message Title
ACCATHRM	X'1230'	4656	Not Authorized to Use Access Method
ACCINTRM	X'1266'	4710	Access Intent List Error
ACCMTHRM	X'1231'	4657	Invalid Access Method
ADDRRM	X'F212'	61970	Address Error
AGNPRMRM	X'1232'	4658	Permanent Agent Error
BASNAMRM	X'1234'	4660	Invalid Base File Name
CHGFATRM	X'1261'	4705	Change File Attributes Rejected
CLSDMGRM	X'125E'	4702	File Closed with Damage
CMDCHKRM	X'1254'	4692	Command Check
CMDCMPRM	X'124B'	4683	Command Processing Complete
CMDNSPRM	X'1250'	4688	Command Not Supported
COMMRM	X'F207'	61959	Communications Error
CSRNSARM	X'1205'	4613	Cursor Not Selecting a Record Position
CVTNFNRM	X'F202'	61954	Conversion Table Not Found
DCLCNFRM	X'1220'	4640	Declare Conflict
DCLNAMRM	X'1256'	4694	Invalid Declared Name
DCLNFNRM	X'1257'	4695	Declared Name Not Found
DDFNFNRM	X'F201'	61953	Data Description File Not Found
DFTRECRM	X'1204'	4612	Default Record Error
DRCATHRM	X'1237'	4663	Not Authorized to Directory
DRCFULRM	X'1258'	4696	Directory Full
DTARECRM	X'1206'	4614	Invalid Data Record
DUPDCLRM	X'1255'	4693	Duplicate Declared Name

Table 3 (Page 2 of 3). DFM Reply Messages

Reply Message ID	Hexadecimal	Decimal	Message Title
DUPFILRM	X'1207'	4615	Duplicate File Name
DUPKDIRM	X'1208'	4616	Duplicate Key Different Index
DUPKSIRM	X'1209'	4617	Duplicate Key Same Index
DUPRNBRM	X'120A'	4618	Duplicate Record Number
ENDFILRM	X'120B'	4619	End of File Condition
EXSCNDRM	X'123A'	4666	Existing Condition
FILATHRM	X'123B'	4667	Not Authorized to File
FILDMGRM	X'125A'	4698	File Damaged
FILERRRM	X'F216'	61974	File Error
FILFULRM	X'120C'	4620	File Is Full
FILIUSRM	X'120D'	4621	File In Use
FILNAMRM	X'1212'	4626	Invalid File Name
FILNFNRM	X'120E'	4622	File Not Found
FILNOPRM	X'1211'	4625	File Not Open
FILOLORM	X'121D'	4637	File Open Lock Option Changed
FILSNARM	X'120F'	4623	File Space Not Available
FILTNARM	X'121E'	4638	File Temporarily Not Available
FUNATHRM	X'121C'	4636	Not Authorized to Function
FUNNSPRM	X'1250'	4688	Function Not Supported
HDLNFNRM	X'1257'	4695	File Handle Not Found
INTATHRM	X'125C'	4700	Not Authorized to Open Intent for Named File
INVFLGRM	X'F205'	61957	Invalid Flag
INVRQSRM	X'123C'	4668	Invalid Request
KEYDEFRM	X'123D'	4669	Invalid Key Definition
KEYLENRM	X'122D'	4653	Invalid Key Length
KEYMODRM	X'1260'	4704	Key Value Modified After Cursor Was Last Set
KEYUDIRM	X'1201'	4609	Key Update Not Allowed by Different Index
KEYUSIRM	X'123F'	4671	Key Update Not Allowed by Same Index
KEYVALRM	X'1240'	4672	Invalid Key Value
LENGTHRM	X'F211'	61969	Field Length Error
MGRLVLRM	X'1210'	4624	Manager Level Conflict
NEWNAMRM	X'124F'	4687	Invalid New File Name
OBJNSPRM	X'1253'	4691	Object Not Supported

Table 3 (Page 3 of 3). DFM Reply Messages

Reply Message ID	Hexadecimal	Decimal	Message Title
OPNCNFRM	X'1242'	4674	Open Conflict Error
OPNMAXRM	X'1244'	4676	Concurrent Opens Exceeds Maximum
PRCCNVRM	X'1245'	4677	Conversational Protocol Error
PRMNSPRM	X'1251'	4689	Parameter Not Supported
RECCNTRM	X'125B'	4699	Record Count Mismatch
RECDMGRM	X'1249'	4681	Record Damaged
RECINARM	X'1259'	4697	Record Inactive
RECIUSRM	X'124A'	4682	Record In Use
RECLENRM	X'1215'	4629	Record Length Mismatch
RECNAVRM	X'126F'	4719	Record Not Available
RECNBRRM	X'1224'	4644	Record Number Out Of Bounds
RECNFNRM	X'1225'	4645	Record Not Found
RSCLMTRM	X'1233'	4659	Resource Limits Reached on Target System
STRDMGRM	X'1268'	4712	Stream Damaged
SUBSTRRM	X'1265'	4709	Invalid Substream
SRCLMTRM	X'F210'	61968	Resource Limits Reached in Source System
SYNTAXRM	X'124C'	4684	Data Stream Syntax Error
TRGNSPRM	X'125F'	4703	Target Not Supported
UPDCSRRM	X'124D'	4685	Update Cursor Error
UPDINTRM	X'124E'	4686	No Update Intent on Record
VALNSPRM	X'1252'	4690	Parameter Value Not Supported
XLATERM	X'F203'	61955	Translation Error

Appendix A. Distributed FileManager Configuration Examples

The examples in this section describe the steps that a system administrator and end user would take before starting a DFM for Windows application to access remote files. This section assumes that Microsoft SNA and/or IBM Personal Communications (IBM PCOMM) for Windows is configured and active.

Note

This section uses the following notation for environment variables: if a directory MYDIR is defined on drive D of the user's workstation, and DFMUSER is set to D:\MYDIR, then:

```
$(DFMUSER)\myfile
```

is equivalent to:

```
D:\MYDIR\myfile
```

These examples explain how:

1. A system administrator creates the global configuration file: *$(DFMADMN)\dfm.rc*.
2. A system administrator creates the formatted global configuration file: *$(DFMADMN)\dfmcfg.dfm*.
3. An end-user can tailor the global configuration file to create *$(DFMUSER)\dfmrc*, a private configuration file.
4. An end-user creates *$(DFMUSER)\dfmcfg.dfm*, the formatted private configuration file.
5. A system administrator creates a shortcut name that can be used by end-users to access remote files. To do this, the system administrator uses the *dfmnet* command.
6. An end user provides remote logon information for server systems using the *dfmlogon* command.
7. DFM applications access files on a server system.

$(DFMUSER) and $(DFMADMN) represent the directory assigned to the DFMUSER and DFMADMN environment variables, respectively.

Creating the Global Configuration File

To begin configuring DFM for Windows for remote file access, the system administrator must create a text configuration file such as the one below using any system text editor. This file should be named *$(DFMADMN)\dfm.rc*.

```
/************************************/
/* Sample DFM configuration file.   */
/************************************/

/*----------------------------------*/
/* Default conversation parameters. */
/*----------------------------------*/
DEFAULT_MAX_SEND_LIMIT(32767);
DEFAULT_MODE_NAME(#INTER);
DEFAULT_SECURITY(PROGRAM);

/*----------------------------------*/
/* Target definitions.              */
/*----------------------------------*/

/* Definition for target: STLMVS3 (the LU alias of the server system. )
   Note:  STLMVS3 uses SNA mode name of #BATCH instead of
          the default: #INTER.
*/
DFM_TARGET (
    TARGET_SYSTEM (STLMVS3)
    MODE_NAME(#BATCH)
);

/*----------------------------------*/
/* Data conversion definitions      */
/*----------------------------------*/

/* Remote files on STLMVS3 that match the pattern
   wmurphy.*.data1, will be converted between the CCSID 951
   on the target system  and CCSID 037 locally. */

 FILE_DESCRIPTOR_MAP (
     TARGET_SYSTEM (STLMVS3)
     TARGET_FILENAME(wmurphy.*.data1)
     BASE_CCSID (000951)
     VIEW_CCSID (000037)
 ) ;

 /* Remote files on STLMVS3 that match the pattern wmurphy.*.data2
    will be converted using the ADL contained in the two ADL files */

 FILE_DESCRIPTOR_MAP (
     TARGET_SYSTEM (STLMVS3)
     TARGET_FILENAME (wmurphy.*.data2)
     BASE_ADL_FILE (D:\adl\data2.base.adl)
     VIEW_ADL_FILE (D:\adl\data2.view.adl)
     VIEW_CPS_FILE (D:\adl\data2.view.cps)
 ) ;
```

Creating the Formatted Global Configuration File

To create the formatted system-wide configuration file the system administrator issues the command:

```
dfmcfg $(DFMADMN)\dfm.rc $(DFMADMN)\dfmcfg.dfm
```

The *dfmcfg* command creates the binary, formatted configuration file: *$(DFMADMN)\dfmcfg.dfm*. Because of its location in the directory specified by DFMADMN, this file becomes the system-wide configuration file. The administrator does not have to run this command again until the configuration file changes.

Tailoring the Configuration File

In this example, an end user tailors the text configuration file to:

- Access a server system not contained in the configuration file that was defined by the system administrator, and
- Use a different buffer size when accessing STLMVS3
- Identify another group of remote files (wmurphy.*.data3) that require data conversion.

This file is named *$(DFMUSER)*\d*fm.rc*.

```
/************************************/
/* Sample DFM configuration file.    */
/************************************/

/*----------------------------------*/
/* Conversation parameter defaults.  */
/*----------------------------------*/
/* Inherit all defaults from the system-wide configuration file */

/*----------------------------------*/
/* Target definitions.              */
/*----------------------------------*/

/* Definition for target: STLMVS3.
   Note:  Need to access STLMVS3 with send limit of 4096 instead of
          the 32769 specified in the system-wide configuration file.
*/
DFM_TARGET (
    TARGET_SYSTEM (STLMVS3)
    MAX_SEND_LIMIT(4096)
);

/* Definition for target: SA27.
   Note:  This target is not defined in the system-wide configuration
          file.  All parameters for this target will be set to the
          defaults from the system-wide configuration file.
*/
DFM_TARGET (
    TARGET_SYSTEM (SA27)
);

/*----------------------------------*/
/* Data conversion definitions      */
/*----------------------------------*/

/* Need to define some additional data conversions not defined in the
   system-wide configuration file.

   Remote files on STLMVS3 that match the pattern wmurphy.*.data3
   will be converted using the ADL contained in the two ADL files.
   These files may be accessed as keyed files indexed on the
   EMPNUM field, as defined in the ADL files.                    */

 FILE_DESCRIPTOR_MAP (
     TARGET_SYSTEM (STLMVS3)
     TARGET_FILENAME (wmurphy.*.data3)
     KEYLIST ( EMPNUM )
     BASE_ADL_FILE (D:\adl\data3.base.adl)
     VIEW_ADL_FILE (D:\adl\data3.view.adl)
     VIEW_CPS_FILE (D:\adl\data3.view.key.cps)
 ) ;
```

Creating a Private Formatted Configuration File

After creating the text configuration file above, you must then create a formatted version:

```
dfmcfg $(DFMUSER)\dfm.rc
```

The *dfmcfg* command creates the formatted configuration file and names it *$(DFMUSER)\dfmcfg.dfm*. You do not have to run this command again until you change the local text configuration file. When accessing remote files, DFM for Windows uses the system parameters defined in *$(DFMUSER)\dfmcfg.dfm* and *$(DFMADMN)\dfmcfg.dfm*. The values in *$(DFMUSER)\dfmcfg.dfm* override those in *$(DFMADMN)\dfmcfg.dfm*.

Associating Target Systems with a Shortcut Name

The system administrator can create a shortcut name that can be used by any user. This shortcut name is kept in the *dfmnet* control file under the directory defined by the **DFMADMN** environment variable. In order to issue a global *dfmnet*, the system administrator must have authority to write to the *dfmnet* control file, *$(DFMADMN)\dfmnet.dfm*, or to create it if it does not exist. The directory assigned to the **DFMADMN** environment variable should be accessible to all users. To create a global shortcut name, the system administrator specifies the **-g** flag on the *dfmnet* command.

```
dfmnet -g \\STLMVS3\project1\ddm  \\admmvs3
```

where:

admmvs3 is the shortcut name.

STLMVS3 is the SNA LU alias name.

\project1\ddm
defines the optional prefix name for the remote target system file.

In this example, you use the shortcut name **admmvs3** within an application to access remote files that have **project1.ddm** as their file name prefix on a DFSMS/MVS DDM target system assigned to the SNA LU alias name, **STLMVS3**.

You can specify additional shortcut names for your “private” use by accessing the *dfmnet* control file under the directory defined by the **DFMUSER** environment variable. In order to issue a local *dfmnet*, you must have authority to write to the *dfmnet* control file, *$(DFMUSER)\dfmnet.dfm*, or to create it if it does not exist. The directory assigned to the **DFMUSER** environment variable is intended for private use. You are responsible for ensuring that access to the directory is within the intended access scope for users. To create a local shortcut name, you do not specify the **-g** flag on the *dfmnet* command as shown:

```
dfmnet \\STLMVS3\project2\ddm  \\usrmvs3
```

where:

usrmvs3 Is the shortcut name.

STLMVS3 Is the SNA LU alias name.

\project2\ddm
Defines the optional prefix name for the remote target system file.

In this example, you can use the shortcut name **usrmvs3** within an application to access remote files that have **project2.ddm** as their file name prefix on a DFSMS/MVS server system assigned to the SNA LU alias name, **STLMVS3**.

Note: (1) The case of STLMVS3 must match the network access software configuration.

(2) The prefix name might be case-sensitive on the server system.

You do not need to issue the *dfmnet* command again until you want to:

- Remove a shortcut name
- Redefine a shortcut name
- Create a new shortcut name

remove or redefine a shortcut name.

You can determine all active shortcut name definitions by issuing the *dfmnet* command with no parameters. This command returns the following:

```
Licensed Materials - Property of IBM
Distributed FileManager for Windows
5639-B92 (c) Copyright IBM Corp. 1997.  All Rights Reserved
Version 2

Private list of shortcut names:
\\USRMVS3 - \\STLMVS3\project2\ddm

Global list of shortcut names:
\\ADMMVS3 - \\STLMVS3\project1\ddm
```

The system administrator and user can remove a global or local shortcut name through the *dfmnet* command. To remove a global shortcut name, the system administrator must specify the **-g** and **-d** flags and the global shortcut name:

```
dfmnet -d -g \\admmvs3
```

To remove a local shortcut name, you must specify the **-d** and the local shortcut name:

```
dfmnet -d \\usrmvs3
```

Providing DFM for Windows with Remote Logon Information

Before running a DFM for Windows application that accesses files on a remote target system, you must provide DFM for Windows with a user ID and password to use on the target system. Issue the *dfmlogon* command, then provide the password when prompted.

```
dfmlogon STLMVS3 WMURPHY
```

You do not have to reissue the *dfmlogon* command until you want to change your remote user ID or password or until you restart your Windows system.

Accessing Remote Files with a DFM for Windows Application

When the above steps are complete, the user (wmurphy) can use a DFM for Windows application to access files on the server system STLMVS3.

Below are examples of specifying remote file names after the steps above have completed (such as creating a formatted configuration file and associating server systems with shortcut names).

The shortcut name assignments are:

\\usrmvs3 is a shortcut name for:

```
\\STLMVS3\project2\ddm
```

\\admmvs3 is a shortcut name for:

```
\\STLMVS3\project1\ddm
```

Table 4. Specifying remote file names

Filename in Application	Target	Remote filename
\\usrmvs3\file.dat	STLMVS3	project2.ddm.file.dat
\\admmvs3\file1.pdse(mbr1)	STLMVS3	project1.ddm.file1.pdse(mbr1)
\\STLMVS3\file1.pdse(mbr1)	STLMVS3	file1.pdse(mbr1)
\\SA27\file2.pdse(mbr1)	SA27	file2.pdse(mbr1)
\\SA27\wmurphy.dir.file	SA27	wmurphy.dir.file
\\STLMVS3\wmurphy.key.data3	STLMVS3	wmurphy.key.data3

Of the above files, the last (*wmurphy.file1.data2*) matches a TARGET_FILENAME descriptor of a FILE_DESCRIPTOR_MAP definition. This match means that DFM would perform data conversion for records in this file.

Sample ADL Conversion Files

The following ADL files would convert a simple record created by an MVS COBOL application to a format readable by a C program running under Windows. For this example, we assume the records reside in files on the target system STLMVS3 with the name **dfm.*.data1**, where * is a wildcard representing zero or more characters.

- The record declaration from the MVS COBOL application.

```
*
****************************************************************
*
 DATA DIVISION.
*
****************************************************************
*
 FILE SECTION.
 FD  PAYROLL1.
 01  EMP-REC.
     05  EMP-NUM   PIC 9(6).
     05  EMP-NAME  PIC A(40).
     05  EMP-SAL   COMP-2.
*
```

- The corresponding ADL description file (which describes the data on the STLMVS3 server system): D:\adl\data1.base.adl

```
/* Sample ADL to describe base COBOL record.  */
DECLARE  BEGIN;
   SEQUENCE BEGIN;
       EMPNUM:     ZONED CCSID(500) PRECISION(6) ZONENC(x'F') ;
       EMPNAME:    CHAR LENGTH(40) CCSID(500);
       EMPSAL :    FLOAT FORM(FH32) ;
   END;
END;
```

- The C structure that a Windows C application could use to access the remote records.

```
typedef struct _EmpRec
{
    short          sEmpNum ;
    char           achEmpName[40] ;
    float          fEmpSalary ;
} EMPREC, *PEMPREC ;
```

- The corresponding ADL description file (which describes the view of the data on the Windows client system): D:\adl\data1.view.adl

```
/* Sample ADL to describe view C record.  */
DECLARE  BEGIN;
   SEQUENCE BEGIN;
       EMPNUM:     BINARY PRECISION(16) ;
       EMPNAME:    CHAR LENGTH(40) CCSID(437);
       EMPSAL :    FLOAT FORM (FB32) ;
   END;
END;
```

- The file descriptor map statement to add to the private configuration file ($(DFMUSER)\dfm.rc) to enable the conversion.

  Note: The TARGET_FILENAME defines the set of files which will undergo data conversion, using the ADL files defined above.

```
/*-----------------------------------*/
/* File conversion for cobol to c    */
/* - We will allow the default naming*/
/*   for the CPS file:               */
/*   D:\adl\data1.view.cps           */
/*-----------------------------------*/
FILE_DESCRIPTOR_MAP (
    TARGET_SYSTEM (STLMVS3)
    TARGET_FILENAME (dfm.*.data1)
    BASE_ADL_FILE (D:\adl\data1.base.adl)
    VIEW_ADL_FILE (D:\adl\data1.view.adl)
) ;
```

Appendix B. Creating a Customized Conversion Plan

DD&C allows you to filter and manipulate the order of data when it is converted from one record format to another. By default, when converting records, all fields in the input record are converted into the corresponding field in the output record. If you want to change the order of fields in the output record, or eliminate some fields altogether, then you must create a custom plan that defines how to map the fields.

The construction of an ADL plan is beyond the scope of this document. See *SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion* for information on creating an ADL plan. The information below describes what restrictions apply to an ADL plan that DFM can use and how to override the default plans.

DFM uses four default conversion plans:

Name	Description
GETPLN	Used to convert records from the base format to the view format
INSPLN	Used to convert records from the view format to the base format
KYGPLN	Used to convert keys from the base format to the view format
KYIPLN	Used to convert keys from the view format to the base format

You can override any or all of the default plans by including a plan of the same name in either the Base or View ADL files. By convention, plans reside in the view ADL file.

DFM default plans use the following index numbers to refer to declarations:

1. Declarations for the input parameters to the plan
2. Declaration for the output parameters from the plan
3. Base record declaration
4. View record declaration
5. Base key declaration (dynamically created by DFM, if the file is keyed)
6. View key declaration (dynamically created by DFM, if the file is keyed)

The plans must use the following data declarations:

```
DECLARE BEGIN ;
    DEFAULT BINARY BYTRVS(TRUE) COMPLEX(FALSE) CONSTRAINED(FALSE)
        RADIX(2) SCALE(0) SIGNED(TRUE) FIT(ROUND) ;
    word:          CONSTANT 31 ;
    ccsid:         SUBTYPE OF BINARY PRECISION(16) SIGNED(FALSE) ;
    record:        ASIS ;              /* Record as an asis string.  */
    inlen:         BINARY PRECISION (word) ; /* Input record length. */
    inccsid:       ccsid ;                   /* Input CCSID tag.     */
    outmaxlen:     BINARY PRECISION (word) ; /* Size of output buffer*/
    outccsid:      CCSID ;                   /* Output CCSID tag.    */
END ;

DECLARE BEGIN ;
    DEFAULT BINARY BYTRVS(TRUE) COMPLEX(FALSE) CONSTRAINED(FALSE)
        RADIX(2) SCALE(0) SIGNED(TRUE) FIT(ROUND) ;
    word:          CONSTANT 31 ;
    outactlen:     BINARY PRECISION (word) ; /* Actual output length.*/
```

The following ADL descriptions of the default plans are used when no corresponding ADL plan is provided in the Base or View ADL files. They can be used as a template for creating custom plans.

DFM Default Get Plan

The following plan is used by DFM to translate records received from the target system from the Base record format to the View record format.

```
GETPLN: PLAN
   (
    "1".inlen:     INPUT,                /* input record LENGTH    */
    "1".inccsid:   INPUT,                /* input record CCSID     */
    "3"."1":       INPUT                 /* input record           */
                   LENGTH("1".inlen)     /* reference to inlen     */
                   CCSID("1".inccsid),   /* reference to inccsid   */
    "1".outmaxlen: INPUT,                /* output record MAXLEN   */
    "1".outccsid:  INPUT,                /* output record CCSID    */
    "4"."1":       OUTPUT                /* output record          */
                   MAXLEN("1".outmaxlen) /* reference to outmaxlen */
                   CCSID("1".outccsid),  /* reference to outccsid  */
    "2".outactlen: OUTPUT                /* output record length   */
   )
   BEGIN;

    /* The record values from the record input buffer are map-   */
    /* ped from their base file definitions to the corresponding */
    /* view file definitions and put in the output record buffer.*/
    "4"."1" <- "3"."1";                  /* assign base to view    */

    /* Return the actual length of the output key value.         */
    "2".outactlen <- LENGTH("4"."1");

   END;
```

DFM Default Insert Plan

The following plan is used by DFM to translate records sent to the target system from the View record format to the Base record format.

```
INSPLN: PLAN
   (
    "1".inlen:     INPUT,                 /* input record LENGTH    */
    "1".inccsid:   INPUT,                 /* input record CCSID     */
    "4"."1":       INPUT                  /* input record           */
                   LENGTH(INSPLN.inlen)   /* reference to inlen     */
                   CCSID(INSPLN.inccsid), /* reference to inccsid   */
    "1".outmaxlen: INPUT,                 /* output record MAXLEN   */
    "1".outccsid:  INPUT,                 /* output record CCSID    */
    "3"."1":       OUTPUT                 /* output record          */
                   MAXLEN("1".outmaxlen)  /* reference to outmaxlen */
                   CCSID("1".outccsid),   /* reference to outccsid  */
    "2".outactlen: OUTPUT                 /* output record length   */
   )
   BEGIN;

    /* The record values from the record input buffer are map-   */
    /* ped from their base file definitions to the corresponding */
    /* view file definitions and put in the output record buffer.*/
    "3"."1" <- "4"."1";

    /* Return the actual length of the output key value.         */
    "2".outactlen <- LENGTH("3"."1");
    END;
```

DFM Default Key Get Plan

The following plan is used by DFM to translate records received from the target system from the Base record format to the View record format.

Note: This plan is only included in the conversion plan space if the FILE_DESCRIPTOR_MAP indicates that the file is keyed.

```
KYGPLN: PLAN
   (
    "1".inlen:     INPUT,                /* input key LENGTH       */
    "1".inccsid:   INPUT                 /* input key CCSID        */
    "5"."1":       INPUT                 /* Input key buffer.      */
                   LENGTH("1".inlen)     /* reference to inlen     */
                   CCSID("1".inccsid),   /* reference to inccsid   */
    "1".outmaxlen: INPUT,                /* output key MAXLEN      */
    "1".outccsid:  INPUT,                /* output key CCSID       */
    "6"."1":       OUTPUT                /* Converted key value.   */
                   MAXLEN("1".outmaxlen) /* reference to outmaxlen */
                   CCSID("1".outccsid),  /* reference to outccsid  */
    "2".outlen:    OUTPUT                /* output key length      */
   )
   BEGIN;

    /* The key values from the input key buffer are mapped from  */
    /* the Base file definitions to the correspond View file     */
    /* definitions and put in the output key buffer.             */
    "6"."1" <- "5"."1";
    /* Return the actual length of the output key value.         */
    "2".outlen <- LENGTH("6"."1");
   END;
```

DFM Default Key Insert Plan

The following plan is used by DFM to translate records sent to the target system, from the View record format to the Base record format.

Note: This plan is only included in the conversion plan space if the file descriptor map indicates that the file is keyed.

```
KYIPLN: PLAN
   (
    "1".inlen:     INPUT,                 /* input key LENGTH       */
    "1".inccsid:   INPUT                  /* input key CCSID        */
    "6"."1":       INPUT                  /* Input key buffer.      */
                   LENGTH("1".inlen)      /* reference to inlen     */
                   CCSID("1".inccsid),    /* reference to inccsid   */
    "1".outmaxlen: INPUT,                 /* output key MAXLEN      */
    "1".outccsid:  INPUT,                 /* output key CCSID       */
    "5"."1":       OUTPUT                 /* Converted key value.   */
                   MAXLEN("1".outmaxlen)  /* reference to outmaxlen */
                   CCSID("1".outccsid),   /* reference to outccsid  */
    "2".outlen:    OUTPUT                 /* output key length      */
   )
   BEGIN;

    /* The key values from the input key buffer are mapped from  */
    /* the view file definitions to the corresponding base file  */
    /* definitions and put in the output key buffer.             */
    "5"."1" <- "6"."1";
    /* Return the actual length of the output key value.         */
    "2".outlen <- LENGTH("5"."1");
   END;
```

Glossary

A Data Language. A language for describing the fields, arrays, and so on of data records in a programming environment so the records can be transparently accessed by other programming environments.

abend. Abnormal end of task.

access method. The part of the DDM architecture which accepts commands to access and process the records of a file.

ADL. A Data Language

alternate index file. A file that has a different key path over a base file. The base file can be a keyed, direct, or sequential file.

API. Application Programming Interface

array. An object consisting of an ordered collection of homogeneous objects mapped onto N dimensions.

attribute. An object that specifies information about another object, such as the length of a character string, field or the date at which a record was last accessed.

CCSID. Coded Character Set Identifier.

CDRA. Character Data Representation Architecture.

character string. A string of bytes containing characters encoded as specified by its CCSID attribute.

data conversion. A set of programs that convert data according to defined data descriptions. For example, characters can be converted from EBCDIC to ASCII, and numeric data can be converted from System /370 packed decimal to IEEE floating point or ASCII character (or vice versa).

data description. Specification of the layout of data. The data description of data stored in a file can be viewed as a file attribute.

data security. The protection of data against unauthorized disclosure, transfer, modifications or destruction, whether accidental or intentional.

data set. The major unit of data storage and retrieval. It consists of a collection of data in one of several prescribed arrangements which is described by control information that the system has access to.

data stream. All data transmitted through a data channel in a single read or write operation.

DBCS. Double-byte character set. A set of characters in which each character is represented by 2 bytes.

DD&C. Data Description and Conversion. An architecture extension to DDM.

DDM. A set of interfaces that gives users access to data files that reside on remote systems connected by a communication network. The DDM interfaces enable an application program to retrieve, add, update and delete data records in a file existing on a remote system. The DDM interfaces can be used to communicate between systems that have different architectures.

DFM client. Translates requests from the source system for access to file data on a remote system into a standard architected DDM request.

DFM server. A DFM component that accepts remote requests to access data and translates the requests into data management requests on the target system.

direct file. A file that is organized so that there is a relationship between the contents of the records and their positions.

Distributed Data Management (DDM). Architecture for accessing distributed data located in files and distributed relational databases.

Distributed File Management (DFM). Strategy for a set of programming facilities that implement the file aspects of the DDM architecture on those systems which represent distributed environments.

intersystem communication. Communication between different systems by means of SNA facilities.

entity. A record or an object.

fixed-point number. An object representing a number whose precision and scale are fixed.

floating-point number. An object representing a number with fixed precision and floating scale.

full path name. The specifications for a file which includes the drive, directory, filename, and file extension.

HLL. High Level Language

independent LU. A logical unit (LU) that is not controlled by a Systems Network Architecture (SNA) host system.

keyed file. A file organization that supports keyed access to the records of the file.

LAN. Local Area Network.

Local Area Network. LAN

LDM. Local Data Management.

local file. A file that resides on the same system as the application program that is accessing it.

logical unit (LU). In Systems Network Architecture (SNA), a port through which an end user accesses the SNA network in order to communicate with another end user.

LU. Logical unit.

mixed-character string. A character string consisting of both SBCS and DBCS characters.

module. A set of data declarations and plans used to convert data.

object. An instance of a type, such as a field of a record or an attribute of a field.

OEM code page. For Windows 95 and Windows NT systems, this is the generic name for any of the various DOS code pages (or character sets). Each code page supports a certain set of languages.

plan. A program for converting data from one representation to another.

protocol. A set of rules to be followed by communication systems.

record. The basic unit of data stored in a file and transferred between DDM source and target servers. An instance of a field or constructor type.

record file. Record files consist of data fields organized into records that can be accessed as a set of bytes.

remote file. A file that resides on a system other than the system where the application program requesting access to the file resides.

Remote Record Access Support. The DFM function that allows VSAM applications to access remote file data.

SBCS. Single-byte character set. A set of characters in which each character is represented by 1 byte.

sequential file. A file in which records are arranged in exactly the same sequence as they were stored into the file.

SNA. Systems Network Architecture.

source system. A system that requests access to data on another system. In a client/server relationship, it is the client system.

Stream Agent. The DDM program responsible for transformation of data between the stream oriented API requests and the DDM byte requests.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

target system. The system that contains data that is being accessed by another system. In a client/server relationship, it is the server system.

target system data. Data considered to be owned and maintained according to the rules and functions prescribed by the data manager on the target system.

Index

E

F

G

I

K

L

M

O

P

R

S

T

U

V

W

We'd Like to Hear from You

SMARTdata UTILITIES for Windows
Distributed FileManager User's Guide

Publication No. SC26-7134-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - IBMMail: USIB2VVG at IBMMAIL

  Be sure to include the following with your comments:
  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

SMARTdata UTILITIES for Windows
Distributed FileManager User's Guide

Publication No. SC26-7134-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

May we contact you to discuss your comments? Yes No

Would you like to receive our response by E-Mail?

_____

Your E-mail address

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Readers' Comments
SC26-7134-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department HHX/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

SC26-7134-00

Cut or Fold
Along Line

Program Number: 5765-548
5765-549
5622-793
5622-794
5639-B92

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-7134-00