

z/OS
2.5

*DFSMS Distributed FileManager Guide
and Reference*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 127.](#)

This edition applies to Version 2 Release 5 of z/OS® (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2021-09-30

© **Copyright International Business Machines Corporation 1993, 2021.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	vii
Tables.....	xi
About this document.....	xiii
Required product knowledge.....	xiii
Referenced documents.....	xiv
z/OS information.....	xiv
How to send your comments to IBM.....	xv
If you have a technical problem.....	xv
Summary of changes.....	xvii
Summary of changes for z/OS Version 2 Release 5.....	xvii
Summary of changes for z/OS Version 2 Release 4.....	xvii
Summary of changes for z/OS Version 2 Release 3.....	xvii
Chapter 1. Introduction to Distributed FileManager.....	1
Terminology Used in This Book.....	1
Introduction to Distributed Data Processing.....	2
Extended Enterprise Data Access.....	2
Client/Server Perspective.....	3
DFSMSdfp Distributed Data Processing Environment.....	4
Distributed FileManager.....	4
Introduction to the Distributed FileManager Environment.....	5
Components of the Distributed FileManager Environment.....	5
Platforms That Support DDM Architecture Implementations.....	6
How Distributed FileManager Works.....	7
How Distributed FileManager DataAgent Works.....	9
Scenarios for Distributed FileManager.....	10
Chapter 2. Accessing Data Sets with Distributed FileManager.....	11
Accessing z/OS Data Sets.....	11
Data Set Requirements.....	11
Data Set Types Supported.....	11
Data Set Types Not Supported.....	12
File Models Supported.....	12
Default File Attributes.....	12
Distributed FileManager Access Functions.....	13
Record Files and Record Access.....	13
Stream Files and Stream Access.....	15
Directories and Directory Access.....	16
Data Set Naming.....	18
Wild Cards.....	18
Wild Card Restrictions.....	18
Using VSAM Data Sets.....	18
REUSE Attribute for VSAM Data Sets.....	19
Nonreusable Attribute.....	19
Reusable Attribute.....	19

Using PDSE and PDS Data Sets.....	20
Special PDSE and PDS Processing Considerations.....	20
Wildcard Processing Exceptions.....	20
Using PDSEs.....	21
Using PDSs.....	21
Coded Character Set Identifiers.....	21
Setting the CCSID Attribute.....	21
Data Conversion.....	22
Associated DDM Attributes.....	23
Listing DDM Attributes.....	23
Propagating DDM Attributes.....	25
Accessing Data Using the DataAgent Parameters.....	26
Using the DFM DataAgent Filename Suffix Parameters.....	26
Using the AGENT(agent_name<,procedure_parameter>).....	26
Using the PARM(agent_parameter_list).....	26
Using the PGM(program_name).....	27
Using the START(job_name<,job_parameters>).....	27
Chapter 3. Customizing z/OS for Distributed FileManager.....	29
What Is In This Chapter?.....	29
Summary of Customizing Tasks.....	29
Interrelationships of Customizing Tasks.....	29
APPC/MVS Customizing Tasks.....	31
Defining PARMLIB Start Parameters for APPC/MVS.....	32
Creating the Distributed FileManager TP Profile.....	33
Creating the APPC/MVS Side Information Data Set.....	34
Defining PARMLIB Start Parameters for the APPC/MVS Scheduler.....	34
VTAM Customizing Tasks.....	35
Defining the Local LU to VTAMLST.....	35
Defining APPC/MVS Logon Mode Entry in VTAMLIB.....	36
Defining LU and Logon Mode on Partner Systems.....	37
Distributed FileManager Customizing Tasks.....	38
Tuning Distributed FileManager Startup Parameters in System PARMLIB.....	38
Activating Distributed FileManager in System PROCLIB.....	41
Verifying PPT Entries for Distributed FileManager.....	41
ACS Routines for Defining Distributed FileManager SMS Classes.....	41
Establishing Distributed FileManager TP Access Security.....	43
Using RACF to Control Access to the Distributed FileManager TP.....	43
Defining the Distributed FileManager TP Profile to RACF.....	43
Defining a TP Administrator to RACF.....	43
Defining a User ID to RACF.....	44
Implementing RACF Access Protection for TP.....	44
Chapter 4. Operating Distributed FileManager.....	45
Starting the Distributed FileManager Environment.....	45
Starting APPC/MVS.....	45
Starting the APPC/MVS Transaction Scheduler.....	45
Starting Up Distributed FileManager.....	45
Triggering the Distributed FileManager DataAgent.....	45
Monitoring Status of Distributed FileManager Conversations.....	46
Using the DISPLAY APPC Command.....	46
Using the DISPLAY ASCH Command.....	46
Controlling Status of Distributed FileManager Conversations.....	47
Deactivating the Distributed FileManager TP.....	47
Stopping a Local LU with the MVS SET Command.....	47
Stopping DFM for z/OS with the MVS CANCEL Command.....	48

Chapter 5. System Samples.....	49
System SAMPLIB Samples.....	49
GDEAPPC.....	49
GDEAPDEF.....	49
GDEASCH.....	50
DFM00.....	51
GDELOGMD.....	51
GDETPDEF.....	53
GDEPRTL.....	54
DFMX0001.....	55
DFMXAGNT.....	56
DFMXSORT.....	61
DFMXSRTI.....	68
DFMQTSO.....	72
DFMXTSOI.....	78
DFM.....	83
Chapter 6. PPT Entries for Distributed FileManager.....	85
Chapter 7. DFMACALL.C sample.....	87
Chapter 8. DDM File Attributes.....	113
Chapter 9. Application Programming Considerations.....	115
Distributed FileManager Implementation.....	115
DDM Record Access File Creation.....	115
Stream File Creation.....	116
File Access Commands Supported by Distributed FileManager.....	116
DDM Record Access Restrictions.....	120
Stream File API Restrictions.....	121
Logon Mode Requirements.....	121
Appendix A. Accessibility.....	123
Accessibility features.....	123
Consult assistive technologies.....	123
Keyboard navigation of the user interface.....	123
Dotted decimal syntax diagrams.....	123
Notices.....	127
Terms and conditions for product documentation.....	128
IBM Online Privacy Statement.....	129
Policy for unsupported hardware.....	129
Minimum supported hardware.....	129
Programming interface information.....	130
Trademarks.....	130
Glossary.....	131
Index.....	143

Figures

1. Extended Enterprise Environment.....	2
2. Client/Server Cooperative Processing.....	4
3. Example of DDM Source—DDM Target Relationships.....	5
4. Distributed FileManager Processing Environment.....	8
5. DDM Source System for z/OS Target Communication Flow.....	9
6. ALTER Command that Marks a Data Set Reusable.....	20
7. IDCAMS ALTER Command.....	22
8. ISMF Data Set List Columns 34-35.....	24
9. LISTCAT Command.....	24
10. IDCAMS LISTCAT Output Showing DDMEXIST and CCSID Fields.....	25
11. Interrelationships of Customizing Tasks for Distributed FileManager.....	31
12. Basic LUADD Definition.....	32
13. TPADD Command Example.....	33
14. VTAM APPL Definition.....	36
15. Logon Mode Table.....	37
16. DFM Member Example.....	41
17. Data Class Routine.....	42
18. Management Class Routine.....	42
19. Storage Class Routine.....	42
20. APPC/MVS Start Parameters.....	49
21. VTAM APPL Definition in VTAMLST.....	50
22. ASCH Start Parameter Statements to Run DFM.....	51
23. Startup Parameters for Distributed FileManager.....	51

24. VTAM Logon Mode Table Part 1 of 2.....	52
25. VTAM Logon Mode Table Part 2 of 2.....	53
26. MVS/APPC Setup for DFM: TP Definition Utility Part 1 of 2.....	54
27. MVS/APPC Setup for DFM: TP Definition Utility Part 2 of 2.....	54
28. VTAM Partner LU Definition to Run DFSMS/DFM.....	55
29. Starting the DFM DataAgent.....	56
30. DFM DataAgent Sample Routine Part 1 of 5.....	57
31. DFM DataAgent Sample Routine Part 2 of 5.....	58
32. DFM DataAgent Sample Routine Part 3 of 5.....	59
33. DFM DataAgent Sample Routine Part 4 of 5.....	60
34. DFM DataAgent Sample Routine Part 5 of 5.....	61
35. DFM DataAgent Sort Sample Part 1 of 7.....	62
36. DFM DataAgent Sort Sample Part 2 of 7.....	63
37. DFM DataAgent Sort Sample Part 3 of 7.....	64
38. DFM DataAgent Sort Sample Part 4 of 7.....	65
39. DFM DataAgent Sort Sample Part 5 of 7.....	66
40. DFM DataAgent Sort Sample Part 6 of 7.....	67
41. DFM DataAgent Sort Sample Part 7 of 7.....	68
42. Sample Installation for the DFMXSORT DataAgent Routine Part 1 of 4.....	69
43. Sample Installation for the DFMXSORT DataAgent Routine Part 2 of 4.....	70
44. Sample Installation for the DFMXSORT DataAgent Routine Part 3 of 4.....	71
45. Sample Installation for the DFMXSORT DataAgent Routine Part 4 of 4.....	71
46. DFM DataAgent Sample Routine (TSO) Part 1 of 7.....	72
47. DFM DataAgent Sample Routine (TSO) Part 2 of 7.....	73
48. DFM DataAgent Sample Routine (TSO) Part 3 of 7.....	74

49. DFM DataAgent Sample Routine (TSO) Part 4 of 7.....	75
50. DFM DataAgent Sample Routine (TSO) Part 5 of 7.....	76
51. DFM DataAgent Sample Routine (TSO) Part 6 of 7.....	77
52. DFM DataAgent Sample Routine (TSO) Part 7 of 7.....	78
53. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 1 of 7.....	79
54. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 2 of 7.....	79
55. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 3 of 7.....	80
56. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 4 of 7.....	80
57. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 5 of 7.....	81
58. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 6 of 7.....	82
59. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 7 of 7.....	83
60. DFM Startup Procedure.....	84
61. PPT Entries for Distributed FileManager.....	85
62. DFM DataAgent Sample Part 1 of 25.....	87
63. DFM DataAgent Sample Part 2 of 25.....	88
64. DFM DataAgent Sample Part 3 of 25.....	89
65. DFM DataAgent Sample Part 4 of 25.....	90
66. DFM DataAgent Sample Part 5 of 25.....	91
67. DFM DataAgent Sample Part 6 of 25.....	92
68. DFM DataAgent Sample Part 7 of 25.....	93
69. DFM DataAgent Sample Part 8 of 25.....	94
70. DFM DataAgent Sample Part 9 of 25.....	95
71. DFM DataAgent Sample Part 10 of 25.....	96
72. DFM DataAgent Sample Part 11 of 25.....	97
73. DFM DataAgent Sample Part 12 of 25.....	98

74. DFM DataAgent Sample Part 13 of 25.....	99
75. DFM DataAgent Sample Part 14 of 25.....	100
76. DFM DataAgent Sample Part 15 of 25.....	101
77. DFM DataAgent Sample Part 16 of 25.....	102
78. DFM DataAgent Sample Part 17 of 25.....	103
79. DFM DataAgent Sample Part 18 of 25.....	104
80. DFM DataAgent Sample Part 19 of 25.....	105
81. DFM DataAgent Sample Part 20 of 25.....	106
82. DFM DataAgent Sample Part 21 of 25.....	107
83. DFM DataAgent Sample Part 22 of 25.....	108
84. DFM DataAgent Sample Part 23 of 25.....	109
85. DFM DataAgent Sample Part 24 of 25.....	110
86. DFM DataAgent Sample Part 25 of 25.....	111
87. Error Messages.....	121

Tables

1. Tunable Parameters in DFM00..... 38

2. DDM File Attributes..... 113

3. DDM Access Method Commands Supported for Distributed FileManager Sequential Files..... 116

4. DDM Access Method Commands Supported for Distributed FileManager Direct Files..... 118

5. DDM Access Method Commands Supported for Distributed FileManager Keyed Files..... 119

6. DDM Access Method Commands Supported for Distributed FileManager Stream Files..... 120

About this document

This document describes the Distributed FileManager component of DFSMSdftp, henceforth called Distributed FileManager or DFM in this book. Distributed FileManager is a Distributed Data Management Architecture (DDM) server implementation on z/OS. Distributed FileManager provides DDM client implementations on heterogeneous systems with remote access to z/OS data. Applications can access z/OS data independent of where the data is located in a distributed network. Using the Distributed FileManager server can improve your ability to access z/OS data from applications running on client systems.

This document introduces you to Distributed FileManager and how it works, what its data set requirements are, and how to customize z/OS for Distributed FileManager support. This book discusses operating the Distributed FileManager environment and working with DDM client implementations on AIX®, i5/OS and DDM client systems. This document also discusses working with DataAgent through the remote DDM application of SMARTdata UTILITIES (SdU).

For information about accessibility features of z/OS and z/OS.e, for users who have a physical disability, see [Appendix A, “Accessibility,”](#) on page 123.

Required product knowledge

To use this book effectively, you should be familiar with the following:

- On z/OS systems:
 - Characteristics of data sets and access methods
 - Using the Storage Management Subsystem (SMS) to manage data sets
 - Utilities that move or copy data sets
 - Working with Virtual Telecommunications Access Method (VTAM®)
 - Using the Interactive Storage Management Facility (ISMF)
 - Working with Advanced Program-to-Program Communications (APPC)
- Distributed data processing:
 - Fundamentals of DDM
 - System Network Architecture (SNA) LU 6.2 protocol for connecting applications
 - Client/server technology
- DDM source implementations (clients):
 - i5/OS operating system:
 - File system
 - i5/OS DDM
 - i5/OS APPC
 - i5/OS control language (CL)
 - Other DDM client systems:
 - Stream- and record-oriented file access
 - Application programming interfaces (APIs)
 - SMARTdata UTILITIES (SdU)
 - Communications Manager or Communications Manager/2

Referenced documents

The following publications are referenced in this book:

Publication Title	Order Number
<i>Character Data Representation Architecture Overview</i>	GC09-2207
<i>Character Data Representation Architecture Reference and Registry</i>	SC09-2190
<i>Distributed Data Management Architecture: General Information</i>	GC21-9527
<i>z/OS DFSMS Access Method Services Commands</i>	SC23-6846
<i>z/OS DFSMSdss Storage Administration</i>	SC23-6868
<i>z/OS DFSMS Using Data Sets</i>	SC23-6855
<i>z/OS DFSMSdfp Utilities</i>	SC23-6864
<i>z/OS MVS Planning: APPC/MVS Management</i>	SA23-1388
<i>z/OS Security Server RACF Security Administrator's Guide</i>	SA23-2289
<i>VTAM Network Implementation Guide</i>	SC31-6434
<i>VTAM Resource Definition Reference</i>	SC31-6438
<i>VTAM Resource Definition Samples</i>	SC31-6414

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, go to [IBM Documentation \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead [“If you have a technical problem”](#) on page xv.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function

If your comment or question is about z/OS itself, submit a request through the [IBM RFE Community](http://www.ibm.com/developerworks/rfe/) (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Documentation function

If your comment or question is about the IBM Documentation functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Documentation Support at ibmdocs@us.ibm.com.

Feedback on the z/OS product documentation and content

If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS DFSMS Distributed FileManager Guide and Reference, SC23-6848-50
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the [IBM Support Portal](http://support.ibm.com) (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy \(www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/ibm-zos-doc-update-policy?OpenDocument\)](http://www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/ibm-zos-doc-update-policy?OpenDocument).

Summary of changes for z/OS Version 2 Release 5

This information contains no technical changes for this release.

Summary of changes for z/OS Version 2 Release 4

This information contains no technical changes for this release.

Summary of changes for z/OS Version 2 Release 3

This information contains no technical changes for this release.

Chapter 1. Introduction to Distributed FileManager

Today's complex data processing environment often requires accessing data that is distributed among many different computer systems. Distributed FileManager (DFM) helps solve the problems of distributed data processing. It offers services for accessing and processing z/OS data from remote computer systems as if the data were local to the remote systems.

The objective of this chapter is for you to understand the concepts of DFM, its benefits, and how it works. To aid your understanding, this chapter includes the following topics:

- Introduction to distributed data processing
- DFSMSdfp distributed processing environment
- Introduction to the DFM environment
- Scenarios for DFM applications.

Terminology Used in This Book

Before beginning the discussion, please acquaint yourself with the following terms:

associated DDM attributes

Associated DDM attributes are z/OS data set attributes that are defined in DDM. Examples of associated DDM attributes are file size, lock options or end-of-file offset for byte-stream files. Associated DDM attributes are not necessarily exclusive to DDM, but can be common to other applications that access the same data sets.

data set/file

Data set and file are used interchangeably throughout this book. Both terms refer to a named collection of data that is treated as a single unit of data storage and retrieval.

Distributed Data Management Architecture

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

Distributed FileManager

Distributed FileManager (or DFM) is an implementation of target (server) support as defined by DDM. DDM permits systems in an extended enterprise that have DDM source (client) capability to access file data on a DDM target z/OS system. See definitions for *source*, *target*, and *extended enterprise*.

extended enterprise

An extended enterprise is a heterogeneous computing environment that often includes both centralized hosts and distributed workstations connected in a network. Gateways within the extended enterprise provide connections to local area networks (LANs). These LANs can serve any computing systems architecture.

local

Local is your reference point when discussing such entities as platforms or applications. For example, when discussing network conversations from the reference point of a z/OS platform, local refers to entities located on that z/OS system. Similarly, when discussing data access methods from the reference point of a z/OS platform, local refers to those access methods. Contrast with *remote*.

partner

Refers to complementary information or function on a remote platform. For example, the ability of DFM to conduct a network conversation requires a local logical unit (LU) on the target z/OS system and a partner LU on the source system.

platform

A computer system running a specific operating system connected in a network. For example, i5/OS and z/OS are different operating system platforms.

record-oriented file

File with a record-oriented structure that is accessed record by record. This file structure is typical of data sets on VM, z/OS, OS/390, and i5/OS systems. Contrast with *stream-oriented file*.

remote

Remote is relative to your reference point when discussing such entities as platforms or applications. For example, when discussing network conversations from the reference point of a z/OS platform, remote refers to entities that access z/OS data across a network. For example, a DDM client application accessing local z/OS data would be remote. Contrast with *local*.

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 client. Source and client are used interchangeably within the scope of this document. Contrast with *target*.

stream-oriented file

File with a byte-oriented structure that is accessed as continuous streams of data bytes. This file structure is common in workstation environments. 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 server. Target and server are used interchangeably within the scope of this document. Contrast with *source*.

Introduction to Distributed Data Processing

Topics included in this introduction are extended enterprise data access, requirements for accessing remote data in a network, and the client/server perspective.

Extended Enterprise Data Access

The extended enterprise environment depicted in [Figure 1 on page 2](#) represents today's data processing installations. Such an environment often includes both centralized hosts and distributed workstations or hosts connected in a network. Gateways provide connections to LANs. These LANs can serve any computing systems architecture.

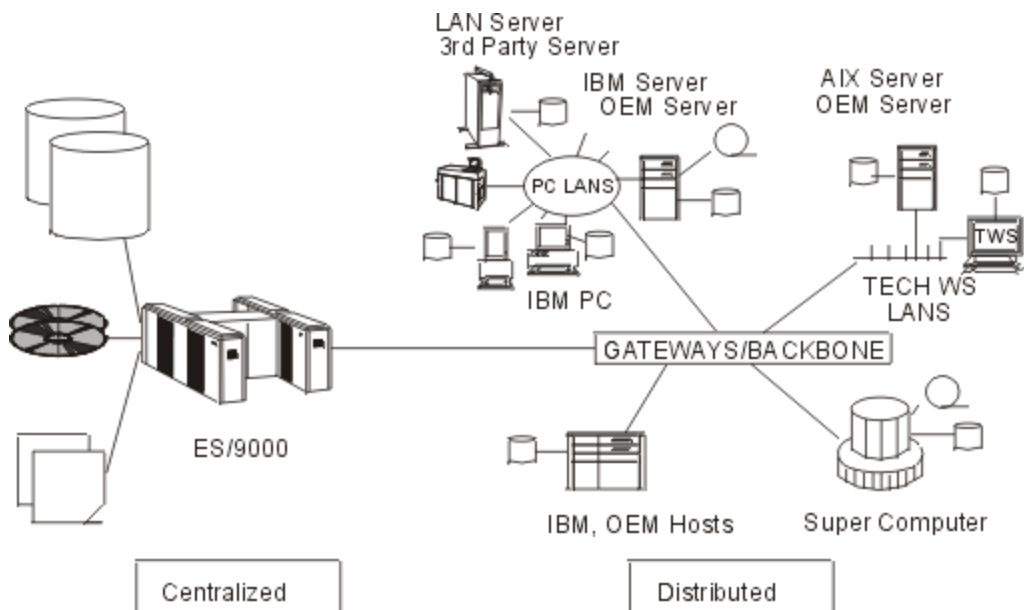


Figure 1. Extended Enterprise Environment

Most applications in an extended enterprise tend to share data to some degree. The trend toward sharing data will grow as workstations become more powerful, networks become more widespread, and applications are written that exploit these capabilities. The evolution of the distributed environment has created the following new requirements for accessing remote data.

Transparent Data Access

Data access should be transparent to applications regardless of the internal format and location of the data. In addition, the data that applications access must at all times be the latest copy. In this discussion, data access implies that applications on source systems can create, read, and write data on target z/OS, or OS/390 systems.

Sharing and Accessing Data

Data in an extended enterprise must be in a form that can be shared throughout an enterprise. Multiple workstations must have access to the same z/OS data in which the z/OS system provides data sharing and serializing at the data set level.

Avoiding Duplicate Data

In an extended enterprise, uncontrolled data duplication leads to storage management problems and wasting of storage resources. In contrast, controlled duplication for backups and migration is desirable and necessary. Downloading data to the local or LAN environment becomes unnecessary for applications with access to z/OS data through DFM services.

Portable Applications

For any computing that is to be off-loaded from mainframe systems to workstations, applications should be readily portable to workstations without downloading data. You should be able to access and share data resident on z/OS systems by running applications on workstations.

Applications developed on workstations before they are ported to z/OS systems should also be able to access data on z/OS systems without downloading it to the workstations or to the LAN servers supporting the workstations.

Transparent Applications

Sometimes pertinent data is spread out in an extended enterprise, some of it local to the workstation where the application is running, and the rest of it on a remote z/OS system. If so, a transparent application that runs without modifications allows existing or new applications to access data wherever the data exists without unnecessary data movement. These applications frequently require both record- and stream-oriented data.

Client/Server Perspective

From an architectural point of view, the client or server can be a workstation, a central processor, a local processor or a departmental processor. Generally, a client is best described as a workstation. It is possible, however, for a large host system to be a client that requests data from a small computer such as a workstation.

Usually a server is a central processor, a local system or a departmental system. It is possible, however, for a workstation to be a server that provides data to a central processor. See [Figure 2 on page 4](#).

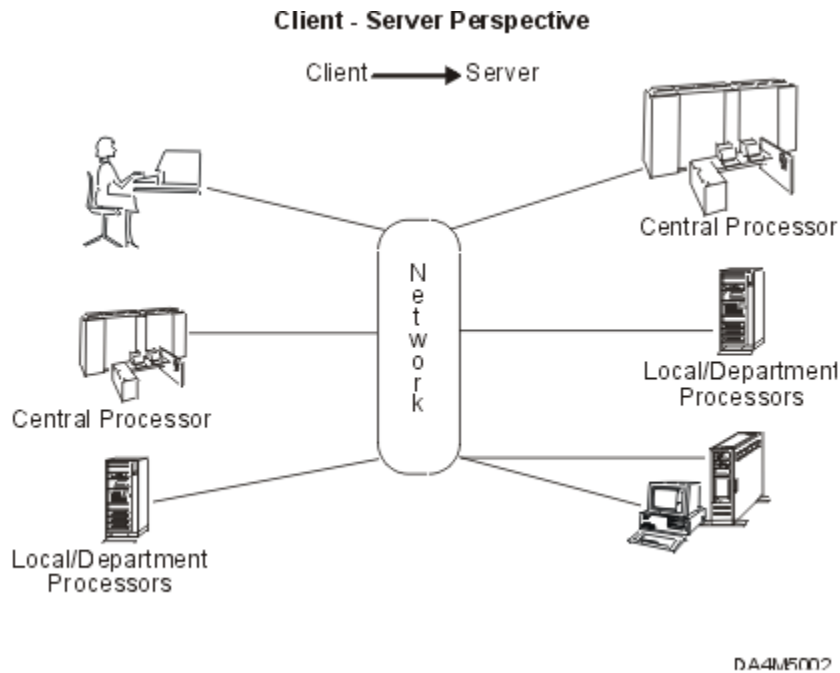


Figure 2. Client/Server Cooperative Processing

DFSMSdfp Distributed Data Processing Environment

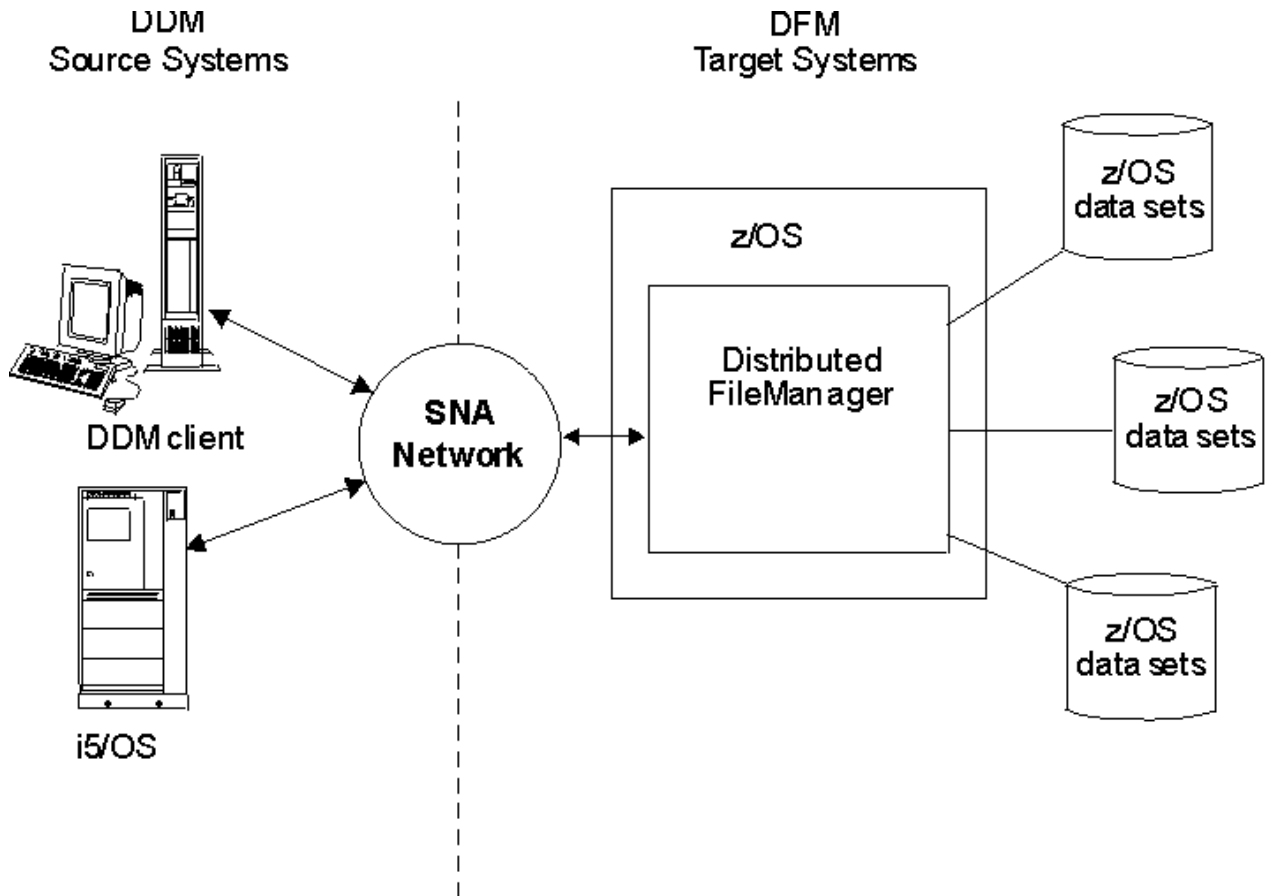
A key objective of DFSMSdfp is to offer products that provide workstations with both record- and stream-oriented data access to z/OS data. Workstations accessing z/OS data must have the capability of creating, reading, and writing data to the z/OS system-managed external storage. DFM is a DFSMSdfp client/server product that enables remote clients in your network to access data on z/OS systems.

Distributed FileManager

DFM is a DDM server on a z/OS system. DDM enables clients to share and access data on z/OS servers regardless of where the data is located. The benefits of DFM are:

- It provides applications and end-users with transparent access to z/OS data from remote platforms
 - Supports both record- and stream-oriented data
 - Gives workstations access to z/OS data as if the data were local
 - Allows you to use local commands; no need to use z/OS commands
- It improves the productivity of application programmers
 - Can develop high-level language applications independent of data location
 - Eliminates upload and download procedures. Data access is in-place
 - Can share data with other workstations as well as with z/OS batch jobs and Time Sharing Option (TSO) users
 - Allows creating, updating, deleting, and renaming of z/OS data that is accessed in-place
- It capitalizes on strengths of centralized data storage
 - Offers backup and recovery support across an extended enterprise
 - Allows data to be shared throughout an extended enterprise
 - Ensures security and data integrity using normal z/OS conventions
 - Provides latest storage and data management techniques for workstation data
- It leverages existing investments in data, applications, support skills, and storage capacity

DFM uses APPC LU 6.2 protocol to establish network conversations with DDM clients. The conversations consist of DDM commands and messages. DDM is the common language between DDM clients and DFM. The DDM client support is currently available on DDM client and i5/OS systems (see [Figure 3 on page 5](#)).



DA4M5003

Figure 3. Example of DDM Source—DDM Target Relationships

Introduction to the Distributed FileManager Environment

This discussion includes the following topics:

- Components of the DFM environment
- Platforms that support DDM implementations
- How DFM works
- How DFM DataAgent works.

Components of the Distributed FileManager Environment

The DFM environment requires DDM, DDM source systems, APPC/LU 6.2 protocol, and Resource Access Control Facility (RACF®) or an equivalent product.

DDM

DDM implementations use DDM commands as their common language for processing remote data access. DDM provides a vocabulary and set of rules for sharing and accessing data among like and unlike computer platforms. It includes a set of standardized file models and access methods that allows users and applications to access remote data without needing to upload and download files.

DFM is a DDM target implementation providing access and sharing of z/OS files to DDM source implementations. DDM source systems use DDM commands to access and share data on DDM target z/OS systems. DDM defines the following terms concerning remote file access:

Source of the request

Initiates requests for data that resides remotely on another system that has DDM target capability.

Target of the request

Processes requests for data initiated by a source system in the network.

For more information about DDM, see *Distributed Data Management Architecture: General Information*.

DDM Source Systems

i5/OS, AIX, and DDM clients exploit DFM services. i5/OS supports both DDM source and DDM target capabilities. For DDM clients and AIX, DDM source systems are available through SMARTdata UTILITIES (SdU); these source systems provide local record file support and DDM source capability.

The DDM client DDM source implementation requires SdU, which supports record file access for applications written in high-level languages that include C, PL/I and IBM Visual COBOL and Visual PL/I. These high-level languages use the record access feature in a transparent manner so that an application can run from the workstation to access remote record data just by recompiling. It supports stream file access for applications written in C. With SdU, stream access is also invoked for files or directories that are the target of the DDM client commands issued for z/OS.

APPC Communications Protocol

DFM uses the APPC LU 6.2 protocol, defined by Systems Network Architecture (SNA), to communicate with DDM source implementations in an extended enterprise network. The APPC LU 6.2 protocol allows systems in a network to communicate on a peer-to-peer basis.

APPC LU 6.2 support on z/OS systems is provided by APPC/MVS, a part of the base control program (BCP) of z/OS and Virtual Telecommunications Access Method (VTAM). DFM is conversant in APPC/MVS LU 6.2 protocols and commands. A DDM source implementation is conversant in APPC LU 6.2 protocols and commands.

APPC LU 6.2 on a DDM source system and LU 6.2 on a z/OS system enable conversations to take place between the DDM source and DFM. These conversations carry DDM commands and messages involved in processing remote access to z/OS data.

For more information about APPC/MVS, see [z/OS MVS Planning: APPC/MVS Management](#).

RACF Conversation Access Security

RACF, or an equivalent product, is used to control which source systems are authorized to initiate conversations with DFM. By setting up RACF resource class profiles, you can define which user IDs or groups are authorized to access DFM services. You can use RACF resource class profiles to define administrators with update authority to authorize access to DFM.

Once a conversation is initiated, DFM uses RACF services to control the actual data access as well. For more details, see [“Using RACF to Control Access to the Distributed FileManager TP”](#) on page 43.

Platforms That Support DDM Architecture Implementations

DDM source or target implementations are supported on the following IBM platforms:

Platform

Implementation

DDM client

DDM source only

AIX

DDM source only

i5/OS

Both DDM source and DDM target

4680 Point-of-Sale

DDM target only

CICS®/DDM

DDM target only (z/OS and VSE)

How Distributed FileManager Works

The objective of this discussion is to explain how DFM generally works on a z/OS system and how remote applications access z/OS data using DFM. Unless otherwise indicated, you can assume that the DDM source implementation is SMARTdata UTILITIES on a DDM client.

Profile of the Distributed FileManager Environment

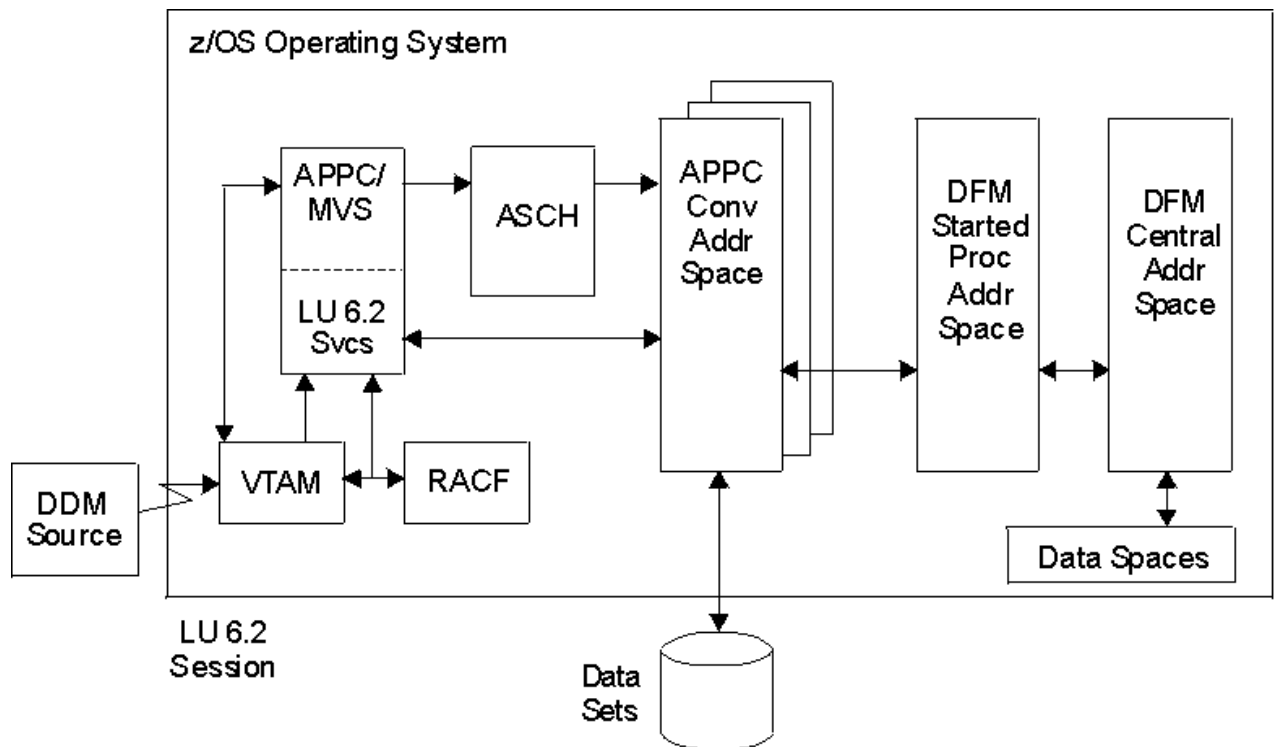
DFM enables authorized users and applications on DDM source systems to access z/OS data remotely. Applications executing on DDM source systems can access z/OS data by exploiting the DFM target support.

DFM participates in APPC LU 6.2 conversations with DDM source implementations. The conversations are exchanges of DDM source commands and DDM target responses. APPC/MVS works with VTAM to provide the logical connection on z/OS for network conversations with source systems. VTAM manages the local logical unit (LU) that forms an LU 6.2 to LU 6.2 link with a partner LU on a remote system.

RACF, or an equivalent product, provides authorization services for controlling access to DFM. Once a conversation is established, RACF also provides authorization services for controlling access to the z/OS data.

DFM provides access to z/OS data to DDM source implementations as follows (see [Figure 4 on page 8](#)):

1. An LU 6.2 to LU 6.2 network link is established between a DDM source and VTAM and APPC/MVS on the DDM target z/OS system.
2. The DDM source sends an LU 6.2 allocate request to initiate a conversation with DFM.
3. The RACF authorizes the DDM source access to DFM.
4. The APPC/MVS scheduler (ASCH), running in an APPC conversation address space, initiates and schedules the DFM started procedure address space.
5. DFM, running in the DFM central address space, processes the allocate request from the DDM source. It begins a network conversation with the DDM source exchanging DDM commands and messages.

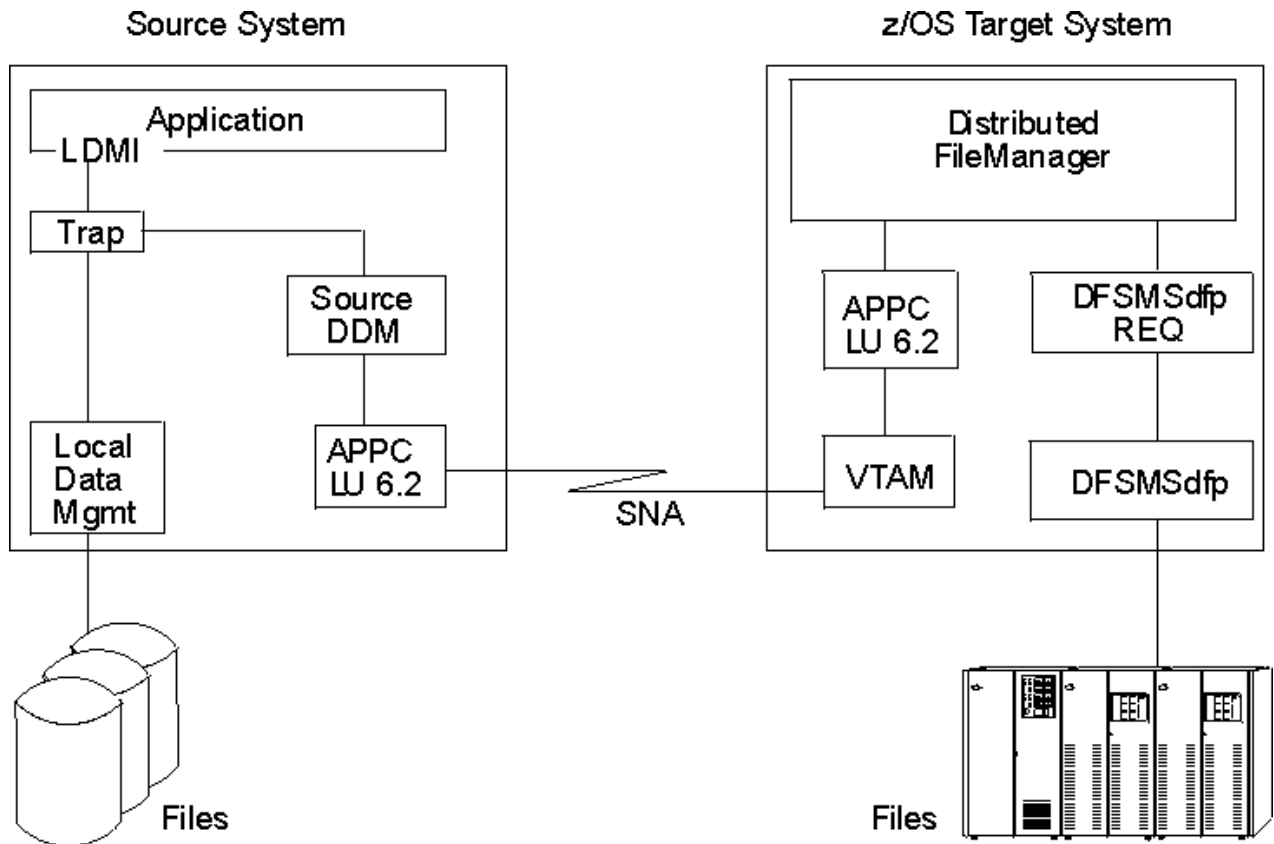


DA4M500 E

Figure 4. Distributed FileManager Processing Environment

How DDM Source Systems Communicate with Distributed FileManager

The communication relationship between a DDM source system and DFM is shown in [Figure 5 on page 9](#).



DA4M5004

Figure 5. DDM Source System for z/OS Target Communication Flow

The DDM source system accesses z/OS data using DFM as follows:

1. An application on the source system requests data.
2. The source LDMI (local data management interface) determines if the data is located locally or remotely.
3. If the data is remotely located, the request is turned over to the source DDM (DDM source).
4. An LU 6.2 to LU 6.2 network link is established between APPC LU 6.2 on the source system and VTAM and APPC LU 6.2 on the z/OS system.
5. Source DDM sends an allocate request across the network link to DFM.
6. If the source is authorized access, a network conversation begins between source DDM and target DFM that processes remote data access to z/OS data.

How Distributed FileManager DataAgent Works

The DFM DataAgent function allows workstation users to invoke remote procedures that run as extensions or agents of DFM. This expands the capability of this mode of access by providing access to functions or data sets not ordinarily supported by DFM and by allowing workstations greater control over processing on the z/OS server.

DFM DataAgent allows the workstation user of SdU to invoke user-written, IBM-written, or vendor-written DFM DataAgent routines using:

- TSO commands, CLISTs, or REXX execs
- DFM DataAgent routines through the remote file access feature of SdU.

The DFM DataAgent is an extension to the DFM component of DFSMSdfp and to the remote DDM application of SdU that provides the ability for remote callers to invoke the DFM DataAgent function to execute specified routines on z/OS. The functions that may be performed using this facility include the execution of TSO and REXX commands as well as user-written programs. Samples are provided that show specific uses of this function. The DFM DataAgent enhancement represents a significant extension of the functionality of the remote DDM application beyond basic data access.

A sample DataAgent is provided to invoke TSO functions, such as TSO CLISTs, REXX execs, or TSO commands. Another sample DataAgent is provided to invoke SORT.

The following is an example of how the DFM DataAgent expands the capability of the remote DDM application on z/OS. Before the DataAgent, the remote application was able to create and delete files, and read, write, update, and delete data contained in files on z/OS. With the DataAgent, this capability is broadened significantly by the ability to execute remote procedures on z/OS. A DataAgent job could be invoked to preprocess data on z/OS by retrieving it from the files or other repositories and place it in the file that the application will access. The DDM application could then process the data in the intermediate file. When the DDM application has finished, a second DataAgent could be invoked to take the data in the intermediate file and distribute the changes to the permanent files.

Scenarios for Distributed FileManager

DFM offers distributed data processing solutions for a broad range of diverse applications. The following are a few scenarios of the many possibilities:

- Insurance industry
 - Customer-written PC applications can present insurance data to underwriters.
 - DDM client and DDM source systems can connect with DFM on a target z/OS system and remotely access and update insurance information.
- Chemical industry
 - Orders can be entered on an i5/OS system.
Source DDM on an i5/OS system can transmit orders to DFM providing access to z/OS data sets for centralized tracking.
 - An i5/OS system contains personnel data for security guards at one establishment. Daily updates of personnel data can be retrieved from an z/OS or OS/390 system using DFM.
- Banking and finance industries
 - Foreign currency transactions on a branch i5/OS can be transmitted to a central z/OS system using DFM.

Chapter 2. Accessing Data Sets with Distributed FileManager

This chapter describes Distributed FileManager (DFM) support for z/OS data sets. It explains the types of data sets you can access and describes the record, stream, and directory access functions you can use. It also includes information about data set naming, data set usage, character sets, and file attributes.

Accessing z/OS Data Sets

This section introduces the data set access capabilities of DFM, including:

- DFM data set requirements
- Data set types supported by DFM
- File models supported by DFM
- Default file attributes

Data Set Requirements

DFM has the following data set requirements:

- New data sets created using DFM should be SMS-managed, although DFM supports non-SMS-managed data set creation and access of existing non-SMS-managed data sets.

Note: The creation of non-SMS-managed data sets is not recommended, because DFM cannot save attributes designed to improve performance or to enhance function. Non-SMS-managed data set creation by DFM should only be used during the transition period between DFM installation and the implementation of system-managed storage. Once this transition is complete the *UNIT* and *VOLUME* parameters should be removed from DFM00. Refer to [“Tuning Distributed FileManager Startup Parameters in System PARMLIB” on page 38](#) for additional information on the tunable parameters in DFM00.

- All data sets accessed must be cataloged in an integrated catalog facility catalog.
- All data sets accessed must reside on direct access storage.
- All data sets must be one of the supported types in the next section.

Data Set Types Supported

DFM supports the following z/OS data set types:

- Non-SMS-managed data sets
- Sequential access method (SAM) data sets
 - Basic sequential access method (BSAM) data sets
 - Queued sequential access method (QSAM) data sets
- Virtual Storage Access Method (VSAM) data sets
 - Entry-sequenced data sets (ESDSs)
 - Key-sequenced data sets (KSDSs)
 - Fixed-length relative record data sets (RRDSs)
 - Variable-length relative record data sets (VRRDSs)
 - VSAM alternate indexes to ESDSs or KSDSs
- Basic partitioned access method data sets
 - Partitioned data set extended (PDSE) members

- Partitioned data set (PDS) members
- Read-only support for PDSE directories
- Read-only support for PDS directories

Data Set Types Not Supported

DFM does not support:

- VSAM linear data sets (LDSs)
- Generation data groups (GDGs) and generation data sets (GDSs)
- Basic direct access method (BDAM) data sets
- Extended format sequential data sets
- z/OS UNIX System Services (z/OS UNIX) hierarchical file system (HFS) files
- Tape media
- z/OS File System (zFS)
- Large format sequential data sets

File Models Supported

The IBM Distributed Data Management (DDM) architecture helps client applications access server data by defining common data access rules that can be used between different kinds of systems. DFM supports a set of standardized DDM file models, that allow client applications to use the DDM architecture to access z/OS data. One or more DDM file models can be used to access each supported z/OS data set listed below:

Record Files

- The DDM direct file model can be used to create and access VSAM RRDSs and VRRDSs.
- The DDM keyed file model can be used to create and access VSAM KSDSs.
- The DDM sequential file model can be used to create and access SAM data sets, VSAM ESDSs, VSAM RRDSs and VRRDSs, PDSE members, and PDS members.
- DFM supports non-SMS-file creation.

Stream files

- The DDM stream file model can be used to create and access SAM data sets and PDSE members. In addition, DFM supports stream access to record files.
- DFM supports non-SMS-file creation.

DFM also supports the following DDM file models, that have no exact equivalent on z/OS:

- A DDM alternate index file (AIF) can be used to create a VSAM alternate index and alternate index path, and to access VSAM data through an alternate index path.
- A DDM directory can be used to access a PDSE directory, a PDS directory, or a directory consisting of all target data sets selected by a source system.

Default File Attributes

Unless otherwise specified by a DDM record access application or changed through the appropriate workstation command, z/OS files have the following default values:

- File hidden, system file, and file protect are set to FALSE.
- Get, insert, and modify capabilities are set to TRUE.

Default Delete Capability Attribute

The default delete capability attribute for the following categories is FALSE, unless you take action to change the delete capability attribute to TRUE: All SMS-managed VSAM KSDSs, RRDSs, and VRRDSs, which were not created through DFM, have a default delete capability of FALSE.

Changing the Delete Capability Attribute

You cannot issue a DDMOpen with delete access intent against data sets that have a delete capability attribute of FALSE.

To change the delete capability attribute for a file, perform either of the following steps:

- Delete the file, and recreate it through a DDM record access application specifying DELCP in the DDMCreate command.
- Add a DDMSetFileInfo command to the application to set the delete capability as desired.

Distributed FileManager Access Functions

This section describes the DFM support for the following categories of files and access functions:

- Record files and record access functions
- Stream files and stream access functions
- Directories and directory access functions

Record Files and Record Access

DFM supports four DDM record file classes, two sets of DDM access methods for record files, and a complete range of DDM record access functions. The file classes, access methods, and access functions you can use depends on the type of z/OS data set you want to access.

Record File Classes

The DDM file classes that you can use to create and access record files are the direct file, keyed file, sequential file, and alternate index file classes. The DDM file classes correspond to the DDM file models explained in [“File Models Supported” on page 12](#).

DDM file classes can be used to access record files as follows:

- SAM data sets, VSAM ESDSs, PDSE members, and PDS members can be accessed through the sequential file class.
Note: SAM data sets and PDSE members can also be accessed through the stream file class, as explained in [“Stream Files and Stream Access” on page 15](#).
- VSAM RRDSs and VRRDSs can be accessed through the sequential file class or the direct file class.
- VSAM KSDSs can be accessed through the keyed file class.
- A VSAM alternate index and associated alternate index path can be accessed through the alternate index file class.

Access to Record Files

You can use DDM record access methods and DDM keyed access methods to access the following record files.

- Record access methods can be used with these record files: SAM data sets; VSAM ESDSs, RRDSs, and VRRDSs; PDSE members and PDS members.
- Keyed access methods can be used with these record files: VSAM KSDSs and DDM AIFs.

- Record and keyed access methods can *only* be used with record files and they cannot be used to access stream files. The stream access method can be used with stream files and with record files. Stream access to record files is described in [“Stream Files and Stream Access”](#) on page 15.

Record Access Functions

The record access functions that can be performed from a remote system, and the data set types that can use with the functions include the following. The list begins with the most restrictive functions that are limited to certain SMS-managed data sets and ends with the most widely available functions that work with any data set supported by DFM.

Modify Attributes

You can modify DDM attributes associated with the following record files:

- SMS-managed SAM data sets on disk
- SMS-managed VSAM ESDSs, KSDSs, RRDSs, or VRRDSs
- SMS-managed PDSE members

Create File

You can create the following record files:

- Non-SMS-managed data sets
- SMS-managed SAM data sets on disk
- SMS-managed VSAM ESDSs, KSDSs, RRDSs, or VRRDSs
- SMS-managed VSAM alternate indexes and alternate index paths
 - An alternate index and alternate index path are created for you when you create a DDM alternate index file (AIF).
 - The VSAM base data set must meet certain requirements, see [“Using VSAM Data Sets”](#) on page 18.
- SMS-managed PDSE members
 - A PDSE will be created for you first, if it does not yet exist.
- PDS members, with these limitations:
 - You can only create PDS members if the PDS already exists. If a new data set is required, a PDSE will be created.
 - SMS-managed PDSs are recommended, but you can also create PDS members in a non-SMS-managed data set.
 - PDS members do not support DDM attributes.

Delete File

You can delete the following record files:

- SMS-managed SAM data sets on disk
- SMS-managed VSAM ESDSs, KSDSs, RRDSs, or VRRDSs
 - If you delete a VSAM base data set with an alternate index, the alternate index will be deleted for you.
- SMS-managed VSAM alternate indexes and alternate index paths
 - These are deleted for you when you delete DDM AIFs.
- SMS-managed PDSE members
 - Even if you delete the last member of a data set, the data set itself will not be deleted.
- PDS members (whether or not they are SMS-managed)
 - Even if you delete the last member of a data set, the data set itself will not be deleted.

Clear File

You can clear the following record files, whether or not they are SMS-managed:

- SAM data sets on disk
- Reusable VSAM ESDSs, KSDSs, RRDSs, or VRRDSs
- PDSE members and PDS members

Other Access Functions

You can use read, write, and positioning functions with the following record files, and you can rename them or retrieve their DDM attributes:

- SAM data sets on disk
- VSAM ESDSs, KSDSs, RRDSs, or VRRDSs (reusable or nonreusable)
- DDM AIFs (and their associated VSAM base data sets)

DDM attributes are retrieved from the VSAM alternate index or VSAM base data set, depending on the attribute.

- PDSE members and PDS members

Note: PDS members and non-SMS-managed data sets do not support their own DDM attributes. If you retrieve DDM attributes, you will receive default values.

Access Restrictions

When using record files, the following data set access restrictions apply:

- Alias names for PDSE and PDS members are not supported by DFM. Only the true names can be used to access a file. Load libraries cannot be handled properly due to loss of link edit attributes.
- When accessing multivolume data sets, backward processing and direct positioning is not supported, some forms of insert processing to the end of the file are not supported, and retrieval and update requests do not work if they span physical volumes.

See [Chapter 9, “Application Programming Considerations,” on page 115](#) for DDM record access restrictions for multivolume data sets.

- If a local z/OS user updates a PDSE member that was created as a sequential file with associated DDM attributes, all the attributes will be lost. Because loss of attributes can cause data conversion and performance problems, local z/OS users should avoid updating PDSE members that are accessed by DFM.
- DFM cannot create or access SAM data sets or PDSE members with fixed record lengths greater than 32,760 or variable record lengths greater than 32,756. DFM cannot create or access VSAM data sets with record lengths greater than 32,760.

Stream Files and Stream Access

DFM supports stream files in SAM data sets or PDSE members. It also supports stream access to record files in additional types of z/OS data sets. This section only describes stream access to stream files and stream access to record files. For information about record files and record access, see [“Record Files and Record Access” on page 13](#).

The stream files and access functions you can use from a remote system and the data set types you can use with them, include:

Stream Files

Two types of z/OS data sets can be accessed using the DDM stream file class, or file model. You can create, rename, delete, modify DDM attributes, and retrieve DDM attributes for stream files in these data set types:

- SMS-managed SAM data sets on disk
- SMS-managed PDSE members

Stream access is provided on some workstation platforms (currently only DDM client) to allow commands to access remote data transparently. For example, the DDM client EPM editor can be used to browse or update anz/OS file in a transparent manner. In addition, the end user on the workstation can specify *TEXT* on the **DFMDRIVE ASSIGN** or **DFMDRIVE SETPARM** commands to activate stream data conversion and to influence the coded character set identifier (CCSID) used to tag newly created host files. See [“Coded Character Set Identifiers” on page 21](#) for information on CCSID and [“Data Conversion” on page 22](#) for information on the **DFMDRIVE ASSIGN** or **DFMDRIVE SETPARM** commands.

Stream files created while the *TEXT* option is in effect will be converted to the target system code page and tagged with the value specified by HOST_CCSID. If HOST_CCSID is omitted, they will be tagged with the CCSID of the target system. See [“Coded Character Set Identifiers” on page 21](#) for information on supported CCSID code pages. Files for which *BINARY* is specified will be tagged with a CCSID of X'FFFF'.

Legacy data sets tagged with the wrong CCSID can be correctly tagged by running **IDCAMS ALTER**. Otherwise, they will have to be retrieved using the target server defined code page as the PC_CCSID or by using *BINARY* processing. For example, a text file that was stored as binary can be restored by retrieving the file from a drive assigned with the *BINARY* parameter and copying it to a drive with the *TEXT* parameter. The reverse is also true when restoring binary files that were stored as text files.

Files that are not tagged with a HOST_CCSID can be retrieved correctly by starting DFM onz/OS with the CCSID parameter in SYS1.PARMLIB(DFM00) set to the CCSID of the file(s) to be retrieved. The DFM00 CCSID value applies to all untagged files for all client access. Alternatively, a HOST_CCSID can be specified on an exception basis by using the workstation HOST_CCSID parameter on the **DFMDRIVE ASSIGN/SETPARM** command. In either case, *TEXT* must be specified to trigger stream data conversion.

Stream Access

You can use the DDM stream access method to read stream data in the following files:

- Record files in nonreusable VSAM ESDSs, and
- Record files in VSAM KSDSs, RRDs, or VRRDs.

You can use the DDM stream access method to read, write, or clear stream data in the following files:

- Stream files in SAM data sets on disk
- Stream files in PDSE members
- Record files in SAM data sets on disk
- Record files in *Reusable* VSAM ESDS data sets
- Record files in PDSE members and PDS members

Access Restrictions

When using stream files, the following data set access restrictions apply:

- As a DFM user, you cannot use record file access methods to access stream files. However, as a local z/OS user, you can use standard DFSMSdfp record access methods to access stream files.
- Alias names for PDSE and PDS members are not supported by DFM. Only the true names can be used to access a file.
- If a local z/OS user updates a PDSE member that was created as a stream file with associated DDM attributes, the attribute extension cell containing those attributes is lost. As a result, the PDSE member loses its stream file properties and assumes a file class of sequential file. At the present time, there is no way to correct this situation.

Directories and Directory Access

DFM supports DDM directories and read-only directory access functions. You can read the directory entries themselves, and their associated DDM attributes, when you list these directories:

- Selected lists of target data sets

- PDSE directories and PDS directories

Note: For record files, the file size shows the number of records in the file. For stream files, the file size shows the number of bytes in the file. (If the file size is unknown to z/OS, a size of 0 is shown.)

DFM does not support the MKDIR command for directories, nor does it support member names that do not comply with z/OS PDSE or PDS naming restrictions. For example, a client file named AFILE could be copied to an z/OS directory, but AFILE.TXT could not.

Renaming within an z/OS directory is possible, but the full z/OS path name must be given. For example, you could perform a RENAME with the following command:

```
RENAME IBMUSER.PDSE(A) IBMUSER.PDSE(B)
```

However, the following RENAME would not work:

```
CD IBMUSER.PDSE (or DFMDRIVE ASSIGN IBMUSER.PDSE)
RENAME A B
```

Selected Lists of Target Data Sets

From a remote system, you can use DFM to list various target data sets as follows:

- You can use a wild card to select a filtered list of target data sets, and you can view the list as if it were a directory. For example, 'userid.*.PAY' lists all the source requester's PAY data sets.
- You can view lists of SAM, VSAM, PDSE, and PDS data sets. A directory can also include files with access restrictions (see [“Access Restrictions”](#) on page 17).

PDSE Directories and PDS Directories

- If you select a PDSE or PDS as a directory, you can select file names that have a wild card in the member name. For example, you can select the following file names:

MEM*, MEM, or null

- If you do not select a directory, you can select file names that have a wild card in the data set name or member name, but not both. For example:

A.*, A.B, A(MEM*), or A(MEM).

- PDSEs, PDSs, and PDS members do not support their own DDM attributes, so default attributes are displayed.

Access Restrictions

A directory can include hidden files, system files, migrated files, or unsupported files; alias names are not shown. When using directories the following access restrictions apply:

- Hidden or system files

When you create files using DFM, you can mark them as hidden files or system files. Later, when you list the directory, you have the option of excluding either of these kinds of files from the list.

- Migrated files

Directory lists will show default attribute values for migrated files, until they are recalled. When they are recalled, they will show their true attribute values.

- Unsupported files

The directory lists all files with the names you selected, regardless of whether the data set type is supported by DFM. If a file is unsupported, it is listed with default attributes. However, you cannot use DFM to access the file itself.

- Alias names

Alias names for PDSE and PDS members are not supported by DFM. Only the true names are shown when you use DFM to list a PDSE or PDS directory.

Data Set Naming

DDM source applications use a file name parameter to specify target data set names on z/OS. If the source file names conform to z/OS data set naming conventions, they can also be used as the target data set names. However, if you want to use source file names which cannot be used on z/OS, you can implement a name mapping function on the source system.

For example, when you create a SAM data set from a DDM client, you can also use the DDM client naming convention on z/OS (an 8-character file name plus a 3-character extension). However, when you create a PDSE member from DDM client, you might need a file name exit on the DDM client to map the DDM client file names to the z/OS data set names. If you are using SMARTdata UTILITIES for a DDM client, the Distributed FileManager component provides a user exit that lets you write a file name mapping program.

Wild Cards

A wild card is a character that can be used to represent zero or more characters or qualifiers in a data set name. DFM supports the use of wild cards with commands that rename or delete files, or that retrieve directory information. For example, you can use wild cards with the DOS commands DELETE, ERASE, or DIR.

The following wild cards are supported by DFM:

%

Represents one and only one character for which a match is not required. For example:

- If the data set name is AB%.XY, then ABA.XY and ABC.XY match.
- If the data set name is A.B(M%), then A.B(MR) and A.B(MS) match.

Represents zero or more characters for which a match is not required. For example:

- If the data set name is A*.XY, then A.XY and ABC.XY match.
- If the data set name is A.B(M*), then A.B(M) and A.B(MEMX) match.

Represents zero or more qualifiers for which a match is not required. For example:

If the data set name is ABC*.Z, then ABC.Z and ABC.X.Y.Z match.

Represents a member name for which a match is not required. For example:

If the data set name is A.B(*), then A.B(MEMX) and A.B(MEMY) match.

Wild Card Restrictions

When using wild cards, the following restrictions apply:

- Wild cards cannot be used to process a group of PDSE or PDS data sets. For example, ABC*(DE) is not allowed. However, wild cards can be used to process a group of PDSE or PDS members, as shown above.
- Only one wild card can be used in each data set name. For example, ABC%E%.XY and AB*.C(E%) are not allowed.
- Wild cards cannot be used in the first character of the data set name. For example, %BCDE.XY and *ABC.D are not allowed.

Using VSAM Data Sets

This section explains how to use alternate indexes to VSAM data sets.

Alternate Index Files

DFM support for VSAM alternate indexes is provided by a DDM file model called an alternate index file (AIF). A DDM AIF provides access to a VSAM base data set (an ESDS or a KSDS), through a VSAM alternate index path. You can define multiple AIFs over a single VSAM base data set. DFM will create an AIF with RECSZ(4086 32600). The RECSZ parameter on the chosen DATACLAS will be ignored by IDCAMS.

You can choose a DDM AIF name with a maximum of 40 characters.

Base Data Sets

You can use DFM to create an alternate index over an ESDS or KSDS base data set. Before you can build an alternate index, the base data set must meet the following requirements:

- DFM requirements:
 - If the base was created using DFM, it must be a KSDS.
 - If it was created by a local z/OS user, it can be an ESDS or a KSDS.
- VSAM requirements:
 - The base data set must contain records.
 - The base data set must be nonreusable. For more information, see [“REUSE Attribute for VSAM Data Sets” on page 19](#).

Access Restriction

After you create an alternate index file or files, you will have more than one access path to the same VSAM base data set. At that point, you can access the base data set directly or through an alternate index path. However, to avoid locking conflicts, you are advised to use only one access path at a time.

DFM uses VSAM data definition name (ddname) sharing, VSAM SHAREOPTIONS, and the DFM lock manager to ensure data integrity. For simultaneous use of more than one access path at a time, consult [z/OS DFSMS Using Data Sets](#) for additional information about VSAM sharing.

REUSE Attribute for VSAM Data Sets

This section explains how to use the REUSE attribute for VSAM data sets. VSAM data sets can be marked nonreusable or reusable. A nonreusable data set cannot be reopened as a new data set. A reusable data set can be used as a new data set each time it is opened, as if it were empty.

Nonreusable Attribute

A VSAM data set must be marked nonreusable before you can build an alternate index over it. This VSAM requirement applies whether or not DFM is used to build the index.

If DFM is used to build the index, it automatically changes the base ESDS or KSDS to nonreusable, before it creates the alternate index. It is also possible for an z/OS user to mark a data set nonreusable with the IDCAMS ALTER command and the NOREUSE parameter.

Reusable Attribute

All VSAM data sets created by DFM are initially reusable. A VSAM data set must be marked reusable before you can use DFM to clear any data set or write to a VSAM ESDS with the stream access method.

You can mark a data set reusable using the IDCAMS ALTER command with the REUSE parameter.

Note:

1. You must be a local z/OS user to use the ALTER command. You cannot turn on the reusable attribute using DFM.
2. If you delete an alternate index file, you must use the ALTER command if you want to make the base reusable again. DFM does not automatically change the base ESDS or KSDS back to reusable.

3. These restrictions are implemented because it is possible for additional indexes to be defined on z/OS, which are not known to DFM.

Figure 6 on page 20 is an example of an ALTER command that marks a data set reusable.

```
//ALTER      JOB      ...  
//STEPS      EXEC     PGM=IDCAMS  
//SYSPPRINT  DD       SYSOUT=A  
//SYSIN      DD       *  
              ALTER -  
                VSAM.DFM.DATASET -  
                REUSE  
  
/*
```

Figure 6. ALTER Command that Marks a Data Set Reusable

For more information, see the ALTER command in [z/OS DFSMS Access Method Services Commands](#).

Using PDSE and PDS Data Sets

DFM supports both PDSEs and PDSs. PDSEs are recommended because they have more capabilities than PDSs.

Special PDSE and PDS Processing Considerations

You can only create PDS members as sequential record files using DFM. DDM client commands create stream files and can be used to create new PDSE members, but cannot be used to create new PDS members.

A DIR command shows PDSEs and PDSs as directories. However, as discussed in “[Directories and Directory Access](#)” on page 16, DFM does not provide full directory support. Also, ambiguities might arise if a PDSE or PDS name matches a prefix name. For example, you may have a PDSE named IBMUSER.DATA and a sequential file named IBMUSER.DATA.SAMFILE.

Therefore, the following rules are provided to help you control PDSE and PDS member access. These examples assume that the user ID is IBMUSER. Note that rules “1” on page 20 and “2” on page 20 take precedence over any of the other rules.

1. A **DFMDRIVE ASSIGN** specifying a PDSE or PDS implies that all the subsequent file references for that drive will be to members until a change directory (CD) command is issued, in which case see rule “2” on page 20 (except that RENAME requires the full z/OS path name).
2. A change directory into a PDSE or PDS implies that the file names that follow will be members. For example, CD "IBMUSER.MYDIR" implies that a reference to file A will be to member A of PDSE or PDS "IBMUSER.MYDIR" (except that RENAME requires the full z/OS path name).
3. Explicit usage of parentheses in a fully qualified name implies a member. For example, "IBMUSER.MYDIR(A)" refers to member A of the PDSE or PDS "IBMUSER.MYDIR".
4. A file name with a "\" preceding the last qualifier implies a member. For example, "IBMUSER.A.B\C" refers to member C of the PDSE or PDS "IBMUSER.A.B" (except that RENAME requires the full z/OS path name with parentheses around the member name).
5. A file name with a "." preceding the last qualifier implies a nonmember. For example, "IBMUSER.A.B.C" refers to a file named "IBMUSER.A.B.C".

Wildcard Processing Exceptions

Note: Wildcard processing does not necessarily follow these rules. For example, COPY S:IMBUSER.PDSE(A*) C:\MYDIR will copy using long filenames IBMUSER.PDSE(A...). Most likely you will want to first issue the command, CD IBMUSER.PDSE, and then the command, COPY A* C:\MYDIR. This will copy using the 1 to 8 character member names only.

Using PDSEs

Some of the advantages to using PDSEs are as follows:

- PDSEs support member-level DDM attributes, whereas member-level attributes do not exist for PDSs.
- PDSE members can contain stream files, whereas PDS members cannot. However, you can use stream access to PDS members that contain record files.
- PDSEs use dynamic space allocation and reclamation, whereas PDSs need to be compressed periodically with the IEBCOPY utility.
- PDSEs are always SMS-managed. PDSs are not necessarily SMS-managed.

Using PDSs

Some of the limitations of using PDSs are as follows:

- PDSs must be compressed periodically using the IEBCOPY utility.
 - Space used by PDS members that are replaced or deleted cannot be reused until the data set is compressed. The more you update a PDS, the more you need to compress it.
 - For more information on IEBCOPY, see [z/OS DFSMSdfp Utilities](#).
- You can only create PDS members if the PDS already exists. If it does not exist, a PDSE and PDSE member will be created instead.
- PDS members do not support their own DDM attributes, so DDM default attributes are assumed.

Coded Character Set Identifiers

DFM supports a DDM attribute called the coded character set identifier (CCSID). The CCSID attribute specifies an identifier registered with the IBM Character Data Representation Architecture (CDRA) of an encoding scheme for coded character set data. The CCSID attribute is a 16-bit number identifying a specific set of encoding scheme identifier, character set identifiers, code page identifiers, and additional coding-related required information that uniquely identifies the coded graphic character representation used. For example, if a file has a CCSID of 437, it is in USA ASCII format. If it has a CCSID of 297, it is in the French EBCDIC format. The meaning of each CCSID is defined in the IBM CDRA. See *Character Data Representation Architecture Reference and Registry* and *Character Data Representation Architecture Overview* for more information.

All single-byte code page conversions supported by CDRA are supported. DFM provides built-in support for data conversions between code pages 500 and 850. CDRA needs to be activated for code page conversions that are outside the DFM built-in support range. The special PC code page values of 0 and 65535 prevent stream data conversion. These values are not valid with the *TEXT* parameter because not providing the PC code page makes it impossible to determine the delimiters that text processing requires. The special host code page value of 65535 (or BINARY) prevents stream data conversion.

Setting the CCSID Attribute

You can set the CCSID attribute locally or remotely. Note that setting the CCSID attribute identifies the character set used by the file, it does not convert the file to that character set.

Setting the CCSID from a Remote System

When you create a new z/OS data set from a remote system, DFM supports the assignment of a CCSID at the time of creation. Also, if a z/OS data set already exists, DFM supports modification of the CCSID from a remote system. You can assign a CCSID to any SMS-managed data set supported by DFM, except a PDS.

Setting the CCSID from a Local System

If a z/OS data set is SMS-managed (and not a PDS), a local system user can run the IDCAMS ALTER command to set or change the CCSID, without using DFM.

Figure 7 on page 22 is an example of the command.

```
//ALTER      JOB      ...  
//STEPA      EXEC     PGM=IDCAMS  
//SYSPRINT   DD       SYSOUT=A  
//SYSIN      DD       *  
              ALTER -  
                USER1.DFM.DATASET -  
                CCSID(X'01F4')  
  
/*
```

Figure 7. IDCAMS ALTER Command

In Figure 7 on page 22, the CCSID parameter sets the coded character set identifier to X'01F4'. For more information, see the ALTER command in [z/OS DFSMS Access Method Services Commands](#).

Data Conversion

Stream Files: DFM offers limited support of data conversion for stream files. DFM APAR OW16828 for DFSMS/MVS 1.3 provides enhancements to end users who install DDM client enhancements, as described in informational APAR II09011. These DDM client end users can now retrieve z/OS stream data and have it converted to the single-byte code page associated with their workstation.

Using commands equivalent to DFM/2 commands DFMDRIVE SETPARM and DFMDRIVE ASSIGN, the DDM client end user can specify z/OS target parameters that will trigger stream data conversion. The z/OS target parameters are as follows:

BINARY

Specifies no stream data conversion. *BINARY* is the default.

TEXT

Specifies stream data conversion and tags new stream file with the workstation CCSID (PP_CCSID). The current workstation CCSID is automatically passed in by the DDM client, but may be overridden by the PC_CCSID parameter. When *TEXT* is specified, the following parameters are also valid:

- *CRLF* maintains record boundaries using carriage return and line feed as the delimiters. *CRLF* is the default.
- *NL* maintains record boundaries using the new line character as the delimiter.
- *LF* maintains record boundaries using the line feed character as the delimiter.
- *NOEOL* does not maintain record boundaries and treats any delimiters or padding as data.

PC_CCSID=dddd

Specifies the workstation CCSID, *dddd* is the decimal CCSID. The CCSID is ignored for *BINARY* processing and new files are tagged with a CCSID of X'FFFF' to indicate they are not converted. The PC_CCSID setting does not affect the retrieval of binary files, only the *TEXT* parameter triggers data conversion.

HOST_CCSID=dddd

Specifies the CCSID used for stream files created on the target system or for legacy data sets with no explicit CCSID defined. *dddd* is a decimal CCSID from 0 to 65535. A CCSID value of 65535 prevents stream data conversion. If omitted, text files are created or retrieved using the current CCSID from SYS1.PARMLIB(DFM00).

The *TEXT* parameter triggers stream data conversion when required and when the combination of CCSIDs is supported by CDRA.

If the file is not tagged with a CCSID and *TEXT* processing is specified by the workstation, legacy files not tagged with a specific CCSID will default to the CCSID as specified in SYS1.PARMLIB(DFM00).

The HOST_CCSID parameter is not used to override an explicit CCSID associated with a file. It is only used to tag new files or to access files that have no CCSID set.

Record Files

DFM does not provide data conversion services for record files. When DFM stores record files on z/OS, the data is stored in the format sent by the source. If a target system application requires a different data format, data conversion can be done by a source application.

For example, if data is sent from a DDM client source system in ASCII format, it requires conversion to EBCDIC before it can be read by a standard z/OS application. This conversion from ASCII to EBCDIC can be done by the conversion utility provided by the Distributed FileManager component of SMARTdata UTILITIES.

Associated DDM Attributes

Associated DDM attributes are z/OS data set attributes that are defined in DDM architecture. Examples of associated DDM attributes are file size, lock options, or end-of-file offset for byte-stream files. Associated DDM attributes are not exclusive to DDM, but can be common to other applications that access the same data sets.

DFM creates associated DDM attributes when it creates new data sets or changes the attributes of existing data sets. When copying, moving, or backing up data sets that have associated DDM attributes, it is important that you use recommended data moving utilities (see [“Propagating DDM Attributes”](#) on page 25).

The remainder of this section explains the applications and commands you can use to determine if a z/OS data set has associated DDM attributes, and the utilities you can use to propagate associated DDM attributes.

Listing DDM Attributes

You can use one of the following tools to determine whether a data set has associated DDM attributes:

- ISMF data set list application
- IDCAMS DCOLLECT command
- IDCAMS LISTCAT command.

However, these tools cannot determine which specific DDM attributes are associated with a z/OS data set, nor the values of the DDM attributes (except for the CCSID attribute).

Using the ISMF Data Set List

With Interactive Storage Management Facility (ISMF), you can use the data set list application to determine whether SAM or VSAM data sets have associated DDM attributes, and the value of the CCSID attribute.

Using the FILTER, LIST, SORT, or VIEW command, select specified data sets and sort on the DDMATTR field in column 34 and the CCSID DESCRIPTION field in column 35. The DDMATTR field indicates whether or not a data set has DDM attributes and the CCSID DESCRIPTION field gives the value of the CCSID. [Figure 8 on page 24](#) is an example of the resulting output.

```

DGTLP11
COMMAND ==>
DATA SET LIST
SCROLL ==> PAGE
Entries 1-6 of 6
Data Columns 34-35 of 35
ENTER LINE OPERATORS BELOW:
LINE OPERATOR DATA SET NAME DDM ATTR CCSID DESCRIPTION
---(1)--- --- (2)--- (34) --- (35)---
DATASET.NUMBER.A YES JAPANESE PC DATA
DATASET.NUMBER.B NO SPANISH PC DATA
DATASET.NUMBER.C YES ID=00255, NO DESC
DATASET.NUMBER.D ---
DATASET.NUMBER.E NO
DATASET.NUMBER.F --- GERMAN PC DATA
----- BOTTOM OF DATA -----
USE HELP COMMAND FOR HELP; USE END
COMMAND TO EXIT.

```

Figure 8. ISMF Data Set List Columns 34-35

For more details, see [z/OS DFSMS Using the Interactive Storage Management Facility](#).

Using the IDCAMS DCOLLECT Command

You can use the IDCAMS DCOLLECT command to determine if SMS-managed data sets have associated DDM attributes and the value of the CCSID attribute. In the DCOLLECT command output, the DCDDDMEX flag indicates if a data set has associated DDM attributes and the DCDCCSID field contains the value of the CCSID attribute.

Using the IDCAMS LISTCAT Command

You can use the IDCAMS LISTCAT command to determine if SAM or VSAM data sets have associated DDM attributes and the value of the CCSID attribute. [Figure 9 on page 24](#) uses the LISTCAT command to generate a report on a data set named IBMUSER.DFMDATA:

```

//LISTCAT JOB
//STEP EXEC PGM=IDCAMS
//*****
//* PURPOSE: LIST A CATALOG AND A CLUSTER
//*****
//SYSPRINT DD SYSOUT=*
//AMSDUMP DD SYSOUT=*
//SYSIN DD *
LISTCAT LVL(IBMUSER.DFMDATA) ALL
/*

```

Figure 9. LISTCAT Command

[Figure 10 on page 25](#) shows the resulting output. The DDMEXIST field contains the value TEXT, indicating that associated DDM attributes exist. And the CCSID field contains the value X'01F4', NLS EBCDIC STANDARD.

```

IDCAMS  SYSTEM SERVICES                                TIME:08:10
12/02/92
PAGE    1
NONVSAM ----- IBMUSER.DFMDATA.TEST
IN-CAT --- SYS1.ICFCAT.VSYS306
HISTORY
  DATASET-OWNER----- (NULL)      CREATION-----1990.016
  RELEASE-----2          EXPIRATION-----0000.000
SMSDATA
  STORAGECLASS -----NORMAL      MANAGEMENTCLASS--PRIMARY
  DATACLASS -----(NULL)        LBACKUP ---1992.296.0129
VOLUMES
  VOLSER-----SYS309      DEVTYPE-----X'3010200E'      FSEQN-----0
ASSOCIATIONS----- (NULL)
ATTRIBUTES
  STRIPE-COUNT----- (NULL)      CCSID-----X'01F4', NLS EBCDIC STANDARD
DDMEXIST TEXT
IDCAMS  SYSTEM SERVICES                                TIME:08:10
12/02/92
PAGE    2
      THE NUMBER OF ENTRIES PROCESSED WAS:
      AIX -----0
      ALIAS -----0
      CLUSTER -----0
      DATA -----0
      GDG -----0
      INDEX -----0
      NONVSAM -----1
      PAGESPACE -----0
      PATH -----0
      SPACE -----0
      USERCATALOG -----0
      TAPELIBRARY -----0
      TAPEVOLUME -----0
      TOTAL -----1
      THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

```

Figure 10. IDCAMS LISTCAT Output Showing DDMEXIST and CCSID Fields

Propagating DDM Attributes

To reliably propagate DDM attributes when moving their associated files, you must use recommended data movers.

For more information on IMPORT and EXPORT, see *z/OS DFSMS Access Method Services Commands*. For more information on DFSMSdss, see *z/OS DFSMSdss Storage Administration*. For more information on IEBCOPY, see *z/OS DFSMSdss Utilities*.

SAM and VSAM Data Sets

You can use the IDCAMS IMPORT and EXPORT commands to copy or move SAM and VSAM data sets. You can use the DFSMSdss data mover to back up, retrieve, or migrate SAM and VSAM data sets. You must use the DFSMSdss data mover if the data sets are managed by DFSMSHsm.

These data movers automatically propagate associated DDM attributes when moving data sets to other volumes or other systems. DDM attributes associated with SAM and VSAM data sets are not propagated in the following situations:

- If you move data sets to a system that does not support DFM
- If you use IDCAMS IMPORT with the INTOEMPTY parameter
- If you use the IDCAMS REPRO command

PDSE Members

You can use the IEBCOPY utility or the DFSMSdss data mover to copy, move, or back up PDSE members. With IEBCOPY you can create unloaded copies directly to tape or disk. You must use the DFSMSdss data mover if the data sets are managed by DFSMSHsm. These data movers automatically propagate

DDM attributes associated with PDSE members. DDM attributes associated with PDSE members are not propagated in the following situations:

- If you copy or move individual records from one member to another
- If the input data set does not completely replace the output data set
- If you move a PDSE member to a system that does not support DFM
- If you convert a PDSE to a PDS
- If you load an unloaded PDSE to a PDS
- If you copy or move a PDSE member to a PDS

If you move or copy a PDS member or a PDS data set to a PDSE, default DDM attributes will be assigned to the resulting PDSE members.

Accessing Data Using the DataAgent Parameters

This section describes accessing data using the DataAgent parameters. A DataAgent can only be started from a DDM application from a client workstation. You can use the DDMOpen function to get a filename suffix that can be used to start DFM DataAgent processing on z/OS. The DDMClose function terminates DataAgent processing. It issues the DDM commands CLOSE and DELDCL, which actually terminates agent processing by invoking the exit with the DELDCL code point in the parameter list, if requested.

Using the DFM DataAgent Filename Suffix Parameters

The DFM DataAgent filename suffix parameters supported by z/OS are the following:

Using the AGENT(agent_name<,procedure_parameter>)

This parameter specifies the name of the agent that is invoked when a file is declared (at DDMOpen) and, optionally, when the file declaration is deleted (at DDMClose). The agent_name specifies the name of a member that must exist in SYS1.PROCLIB. Parameters can optionally be provided for symbolic substitution in the PROCLIB member.

Allocation will run under the authorization assigned to started tasks. The agent (running under the user's authorization) may have to use dynamic allocation to allocate files that cannot be allocated by started tasks.

Because agents begin as started tasks, unless DFM00 specifies RESTRICT_START(NO), the first 3 characters of the procedure (or agent) must be "DFM."

The maximum length of the agent name and its parameters is 107 bytes. Each parameter in the list of procedures is subject to the z/OS limit of 44 bytes.

The agent runs synchronously. If the PROCLIB member has multiple steps, any file name changes or return code settings will be propagated to the later steps and will only be returned to DFM after the last step has executed.

Using the PARM(agent_parameter_list)

This parameter is used to pass parameters to the agent routine when it begins to execute in the DataAgent address space. Parameters are converted to upper case and concatenated to any parameters provided in the PROCLIB member.

The maximum length of the parameters in bytes is limited only by the space available to the filename suffix.

If PARM is specified, the PROCLIB member must contain the JCL statements as provided by the sample DFMX001 that specifies the DFMINIT parameter so as to run a DFM DataAgent address space initialization routine as the first program in the address space. This causes DFM to pass a supplementary run time parameter list to the DataAgent routine and allows the routine to return an error code and

additional reason codes to DFM. The agent parameter list specified is concatenated with DFMINIT before the DataAgent routine invoked.

This parameter is ignored if AGENT is omitted.

Using the PGM(program_name)

This parameter specifies the name of the program (DataAgent routine) to be invoked by DFM after initialization. If omitted, the program invoked will default to the agent name requiring that you have identically named z/OS load modules and PROCLIB members.

This parameter is ignored if AGENT is omitted.

Using the START(job_name<,job_parameters>)

This parameter specifies the name of the PROCLIB member representing a job or procedure to be started asynchronously. Unless DFM00 specifies RESTRICT_START(NO), the first 3 characters of the procedure or command must be "DFM."

Optional parameters can also be provided. The z/OS limit for the total length of the job name and its parameters is 124 bytes.

DFM will verify that an address space for running the procedure was created, but will not verify that the procedure exists or that it ever completes successfully. That is, the started job runs asynchronously.

It is possible to run some existing PROCLIB members that may not have particular initialization requirements by using only the AGENT keyword. However, it is not recommended because return codes will not be passed back to DFM. It is expected that there will usually be a need for extended parameter passing. The AGENT parameter should be used in conjunction with the PARM and PGM parameters even if the PARM parameter is the null value of PARM() or the PGM name is the same as the agent name.

DFM imposes a limit of 256 bytes for the file name and file name suffix and for the total length of the parameters (AGENT, PARM, PC_CCISD, START, and so on) that can be passed.

As with the other filename suffix parameters, unidentified or misspelled keywords are ignored and the first (leftmost) is used in case of duplicate keywords.

Chapter 3. Customizing z/OS for Distributed FileManager

This chapter is about customizing z/OS for Distributed FileManager (DFM). It discusses how to enable DFM to function in a network as a DDM target (server) providing remote access to data sets for DDM source implementations (clients). DFM does not support DDM source capability.

What Is In This Chapter?

Customizing z/OS for Distributed FileManager includes several tasks. These tasks involve establishing APPC/MVS, VTAM, DFM, and other system information so that DFM can provide remote access to data sets. The objective of this chapter is for you to understand the tasks involved and how they are interrelated.

Summary of Customizing Tasks

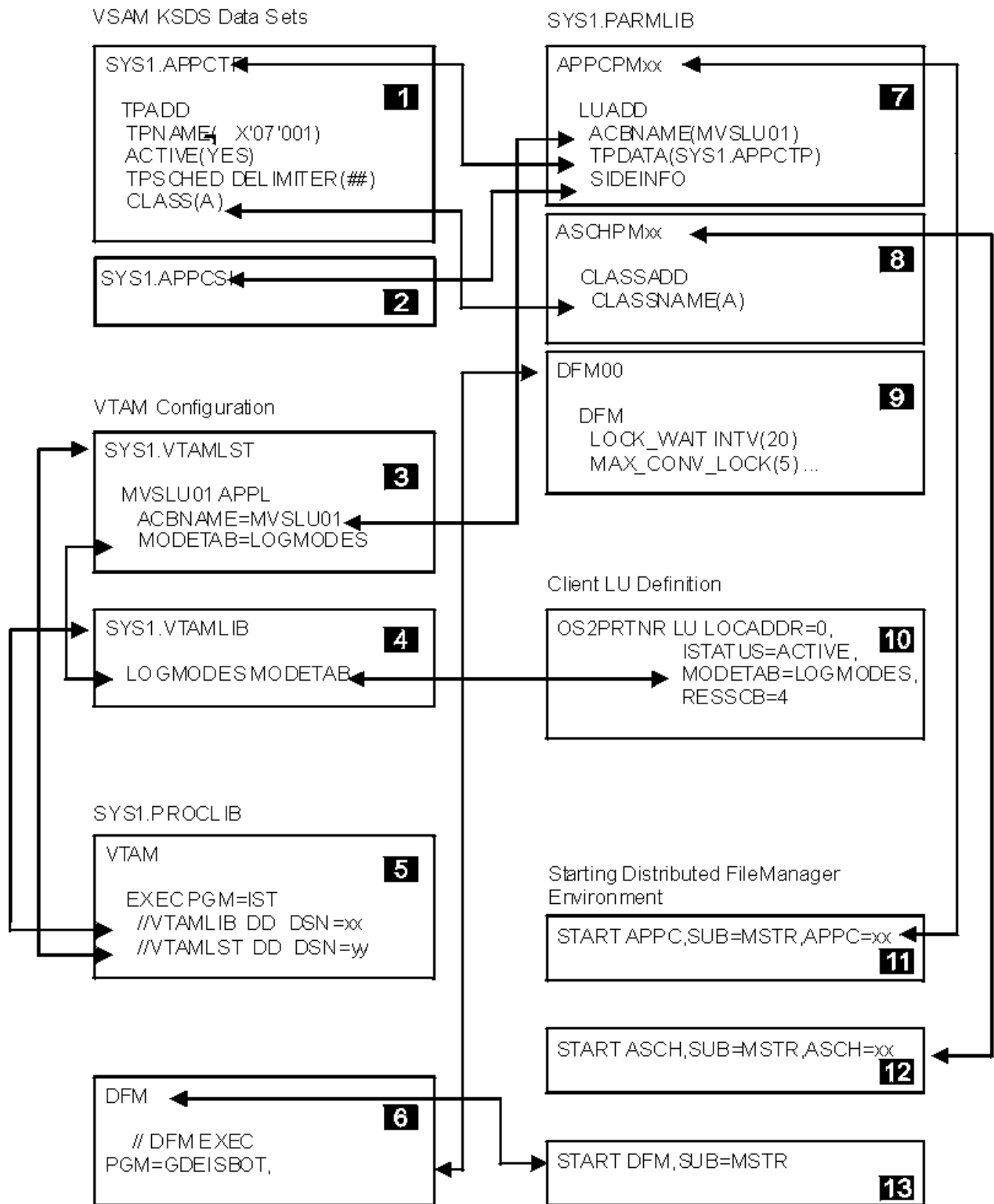
- APPC/MVS customizing tasks
 - Defining PARMLIB start parameters for APPC/MVS
 - Creating the APPC/MVS transaction program (TP) profile data set (if not already existent)
 - Adding DFM TP profile information to the TP profile data set
 - Creating the APPC/MVS side information data set
 - Defining PARMLIB start parameters for the APPC/MVS scheduler
- VTAM customizing tasks
 - Defining the local LU to VTAMLST
 - Defining the logon mode table in VTAMLIB
 - Defining the local LU and logon mode on a partner system
- DFM customizing tasks
 - Installing PARMLIB start parameters for DFM
 - Activating the PROCLIB startup procedure for DFM
 - Verifying program property table (PPT) entries for DFM
- Setting up automatic class selection (ACS) routines
- Defining TP access security

Interrelationships of Customizing Tasks

Figure 11 on page 31 shows some of the interrelationships among tasks involved in customizing z/OS for DFM. Each numbered box in the figure represents a task. To simplify the diagram, only sample parameters that show relationships are shown. The lines and arrows between boxes show relationships between parameters, members, or data set names.

- 1** Adding the DFM TP profile to the TP profile data set
- 2** Creating the APPC/MVS side information data
- 3** Defining a local LU and logon mode table to VTAMLST
- 4** Defining logon mode table to VTAMLIB
- 5** Starting VTAMLST and VTAMLIB
- 6** Startup procedure for DFM

- 7** Defining startup parameters for APPC/MVS
- 8** Defining startup parameters for the APPC/MVS transaction scheduler
- 9** Tunable startup parameters for DFM
- 10** Defining partner DDM client local LU and logon mode information
- 11** Operator command for starting APPC/MVS
- 12** Operator command for starting the APPC/MVS transaction scheduler
- 13** Operator command for starting DFM



DA4M5010

Figure 11. Interrelationships of Customizing Tasks for Distributed File Manager

APPC/MVS Customizing Tasks

APPC/MVS customizing tasks include:

- Defining PARMLIB start parameters for APPC/MVS
- Creating the APPC/MVS TP profile data set (if not already existent)
- Creating the DFM TP profile
- Creating the APPC/MVS side information data set
- Defining PARMLIB start parameters for the APPC/MVS transaction scheduler

Defining PARMLIB Start Parameters for APPC/MVS

You define APPC/MVS start parameters in system PARMLIB member APPCPMxx (for example, SYS1.PARMLIB(APPCPMxx)). The APPC/MVS start parameters contain information for establishing and controlling APPC conversations on the system. They identify the local LU to be used for APPC conversations, and the TP profile and side information data sets to be used by APPC/MVS.

On z/OS, Distributed FileManager conversations flow over the LU defined as the base LU in APPCPMxx. The TP profile data set provides APPC/MVS with the DFM TP profile information. The TP profile information enables DFM to participate in APPC LU 6.2 conversations.

You can control APPC/MVS start parameters by using different versions of APPCPMxx. Each version can have different values for the start parameters. For example, one version can omit an LU name that is included in another version.

APPC/MVS is started by a system operator command. The operator command (which can be part of initial program load) identifies APPCPMxx (see [“Starting APPC/MVS” on page 45](#)). Both APPC/MVS and the APPC/MVS transaction scheduler (also started by operator command) must be active before DFM LU 6.2 conversations can take place.

Using the APPC/MVS LUADD Definition

Use the APPC/MVS LUADD definition to define the start parameters in APPCPMxx. With LUADD parameters, you can add, modify, and delete LU information. You can also change the defined names for both the TP profile and the side information data sets.

Figure 12 on page 32 is an example of the basic LUADD definition that should be included in APPCPMxx. This example can be found in system SAMPLIB member GDEAPPC (for example, SYS1.SAMPLIB(GDEAPPC)), or see [“GDEAPPC” on page 49](#). Also see system SAMPLIB members APPCPMRX and APPCPMXX for more details.

```
LUADD
  ACBNAME(MVSLU01 )      ← LU name
  BASE
  TPDATA(SYS1.APPCTP)    ← APPC/MVS VSAM TP data set
  SIDEINFO DATASET(SYS1.APPCSI) ← APPC/MVS VSAM SI data set
```

Figure 12. Basic LUADD Definition

The parameters in [Figure 12 on page 32](#) do the following:

ACBNAME(MVSLU01)

Defines the name of the LU as MVSLU01. The name specified must be the same as the LU name specified to VTAM.

BASE

Indicates that the LU specified for ACBNAME (MVSLU01 in this case) is the base LU for APPC/MVS. The base LU is associated with the APPC/MVS transaction scheduler. APPC LU 6.2 conversations take place across the base LU.

TPDATA(SYS1.APPCTP)

Defines the name of the TP profile data set as SYS1.APPCTP. The name specified must match the name of the APPC/MVS TP profile data set.

SIDEINFO DATASET(SYS1.APPCSI)

Defines the name of the VSAM KSDS side information data set as SYS1.APPCSI. The name specified must match the name of the APPC/MVS side information data set.

The LUADD SCHED parameter (omitted in the above example) specifies the name of the APPC/MVS transaction scheduler. If omitted, it defaults to the value SCHED(ASCH).

For more information on using the LUADD definition, see [z/OS MVS Planning: APPC/MVS Management](#).

Creating the Distributed FileManager TP Profile

APPC/MVS enables DFM to communicate across a computer network with DDM source implementations. To use APPC/MVS services, DFM must be set up as an APPC/MVS TP. Every TP must have a TP profile contained in the APPC/MVS TP profile data set. The TP profile consists of scheduling and security information needed to run the TP.

Allocating a VSAM KSDS for the TP Profile

If you are not already using APPC/MVS, you must allocate a VSAM KSDS in the system library (for example, SYS1.APPCTP) where APPC/MVS TP profile information can be stored. The name of the VSAM KSDS data set must match the name defined in system PARMLIB member APPCPMxx. A sample of allocating the VSAM KSDS is in system SAMPLIB member ATBTPVSM (for example, SYS1.SAMPLIB(ATBTPVSM)).

For more information on allocating a VSAM KSDS, see [z/OS DFSMS Access Method Services Commands](#).

Adding the TP Profile to the VSAM KSDS

This discussion assumes that a VSAM KSDS already exists for TP profile information.

The APPC/MVS administration utility (ATBSDFMU) has commands for creating and modifying APPC/MVS TP profiles in the VSAM KSDS. You can use the TPADD command to add a TP profile. For a sample of using TPADD to add a variety of TP profiles, see system SAMPLIB member ATBUTIL (for example, SYS1.SAMPLIB(ATBUTIL)).

Figure 13 on page 33 shows a job step that adds a DFM TP profile to SYS1.APPCTP using the TPADD command. This example can be found in system SAMPLIB member GDETPDEF (for example, SYS1.SAMPLIB(GDETPDEF) and in “GDETPDEF” on page 53.

```
//STEP      EXEC  PGM=ATBSDFMU
//SYSPRINT DD   SYSOUT=*
//SYSSDOUT DD   SYSOUT=*
//SYSSDLIB DD   DSN=SYS1.APPCTP,DISP=SHR
//SYSIN      DD   DATA,DLM=XX
TPDELETE
TPNAME(^X'07'001)
TPADD
TPNAME(^X'07'001)          <- TP key section
ACTIVE(YES)                <- TP attribute section
TPSCHED_DELIMITER(##)      <----- TP scheduler section
CLASS(A)
JCL_DELIMITER(ENDJCL)
//GDEDFM JOB MSGCLASS=H,MSGLEVEL=(1,1),CLASS=A
//GDEDFM EXEC PGM=GDEISASB
/* CHANGE THE STEPLIB STATEMENT AS REQUIRED IF YOUR INSTALLATION
/* DOESN'T HAVE THE LE RUNTIME DATA SET IN ITS LINK LIST.
/*STEPLIB DD DSN=SYS1.SCEERUN,DISP=SHR
/*CDRATRC DD DSN=SYS1.CDRATRC2,DISP=SHR <- CDRA API TRACE OUTPUT
/*SYSOUT DD DSN=SYS1.CDRAOUT2,DISP=SHR <- C RUNTIME MESSAGES
//SYSOUT DD DUMMY <- C RUNTIME MESSAGES (NO-OP)
ENDJCL
##
XX <-----
```

Figure 13. TPADD Command Example

As [Figure 13 on page 33](#) shows, each TP profile contains three sections.

TP profile key section

Consists of a TP name and a TP level (because TP level is not included in this example, TP Level defaults to SYSTEM).

- The TP name for DFM must be ^X'07'001
- TP level identifies which entities RACF authorizes to access the DFM TP. It can be one of the following:

Level

Access

SYSTEM

Any user can attach the TP. This is the default if no TP level is used.

GROUP

Any member of a predefined group of users can attach the TP.

USER

A single user can attach the TP.

TP attributes section

Consists of ACTIVE (YES), which indicates that the TP status is active. If the status is set to ACTIVE (NO), the TP cannot be scheduled.

TP scheduler section

Has the following information:

- Provides the JCL used to run the DFM TP. This example shows sample JOB and EXEC statements.
- SCHED(A) indicates the APPC/MVS transaction scheduler for the DFM TP.
- GDEDFM should have either no region size or a region size of 0K to contain cached stream files.
- GDEISASB must be specified as the program to be executed on the EXEC statement.

Note: The language environment (LE) is required to use CDRA. If LE is installed and is not in the link list, SYS1.PROCLIB(DFM) and SYS1.SAMPLIB(GDETPDEF) should be modified so their STEPLIB DD statements refer to the proper LE run time library. Refer to DFMREADM in SYS1.SAMPLIB for details. SYSOUT and CDRATRC files can be allocated as RECFM=FBA, LRECL=133, and DSORG=PS for use in diagnosing CDRA problems.

TP profile definition parameters not included in this example are set to default values. For more details about adding and modifying TP profile information, see [z/OS MVS Planning: APPC/MVS Management](#).

Creating the APPC/MVS Side Information Data Set

APPC/MVS requires that a VSAM KSDS data set be allocated for the side information data set. The APPC/MVS side information data set contains translations of symbolic destination names used by TPs when they issue outbound allocate requests.

Because DFM is a DDM target only and therefore does not issue outbound allocate requests, the side information data set does not need to contain any information. The name used for the SIDEINFO parameter in system PARMLIB member APPCPMxx must match the side information data set name.

Sample JCL for allocating the side information KSDS data set can be found in system SAMPLIB member ATBSIVSM (for example, SYS1.SAMPLIB(ATBSIVSM)). For information about creating side information, see [z/OS MVS Planning: APPC/MVS Management](#).

Defining PARMLIB Start Parameters for the APPC/MVS Scheduler

The APPC/MVS transaction scheduler (ASCH) initiates and schedules TPs in response to inbound requests for conversations. Start parameter values in system PARMLIB member ASCHPMxx (for example, SYS1.PARMLIB(ASCHPMxx)) define and modify TP scheduling classes and other TP scheduling characteristics to ASCH.

ASCH is started by a system operator command. The operator command (can be part of initial program load) identifies the ASCHPMxx member (see [“Starting the APPC/MVS Transaction Scheduler”](#) on page

45). Both APPC/MVS (also started by operator command) and ASCH must be active before DFM LU 6.2 conversations can take place.

You can use an APPC/MVS CLASSADD definition to define the ASCH start parameters in ASCHPMxx, as shown in the following example. This example can be found in system SAMPLIB member GDEASCH (for example, SYS1.SAMPLIB(GDEASCH)) or in “GDEASCH” on page 50.

```
CLASSADD CLASSNAME(A)
      MSGLIMIT(1000) MAX(10) MIN(1) RESPGOAL(1)
```

The parameters in this example do the following:

CLASSADD CLASSNAME(A)

Defines a class of transaction initiators to ASCH. A transaction initiator is an entity, such as DFM, that functions as an APPC/MVS TP. TPs in the same class should have similar characteristics such as run-time, priority, schedule-type, and security.

The CLASSNAME parameter defines the class name as A. It must match the class name used in the TP profile for DFM.

MSGLIMIT(1000)

Defines 1000 as the maximum number of messages in the message log data set for TPs in this class.

MAX(10)

Defines 10 as the maximum number of transaction initiators allowed for this class.

MIN(1)

Defines 1 as the number of transaction initiators in this class that will be started when ASCH is started.

RESPGOAL(1)

Defines 1 second as the system response time goal for TPs running in this class.

For more information about defining ASCH and about scheduling TPs, see [z/OS MVS Planning: APPC/MVS Management](#).

VTAM Customizing Tasks

Customizing VTAM for DFM includes defining the local LU to VTAMLST, setting up the logon mode table in VTAMLIB, and establishing local LUs and logon mode definitions on partner systems.

Defining the Local LU to VTAMLST

When VTAM is initialized on an z/OS system, local LUs are activated based on information in the system VTAMLST library. To define the local APPC/MVS LU to VTAM, use a VTAM application (APPL) definition in the VTAMLST library that is defined in the VTAM start procedure. The VTAM APPL definition:

- Names the local APPC/MVS LU
- Sets up defaults for the LU
- Specifies the name of the logon mode table that contains the logon mode used by the LU
- Defines security for the LU

Figure 14 on page 36 is an example of a VTAM APPL definition. This example can be found in system SAMPLIB member GDEAPDEF, for example, SYS1.SAMPLIB(GDEAPDEF), or see “GDEAPDEF” on page 49.

```

MVSLU01 APPL      ACBNAME=MVSLU01, ← ACBNAME (also
APPC/MVS LUADD)
APP C=YES,
AUTOS E=0,
DDRAINL=NALLOW,
DMINWNL=5,
DMINWNR=5,
DRESPL=NALLOW,
DSESLIM=10,
LMDENT=19,
MODETAB=LOGMODES, ← VTAM Logon Mode Table name
PARSESS=YES,
SECACPT=CONV,
SRBEXIT=YES,
VPACING=1

```

Figure 14. VTAM APPL Definition

The ACBNAME and MODETAB parameters in this example do the following:

ACBNAME=MVSLU01

Defines the local APPC/MVS LU name as MVSLU01. The LU name specified must match the local LU name defined in PARMLIB member APPCPMxx.

MODETAB=LOGMODES

Defines the name of the logon mode table as LOGMODES. This parameter is optional. Including it, however, allows you to make additional logon mode definitions known to VTAM. This is required if the logon mode name specified by the partner system is not supplied in the VTAM default logon mode table ISTINCLM.

The name for the logon mode table name must match the name of a defined in VTAMLIB.

For more information about VTAM APPL definitions, see [z/OS MVS Planning: APPC/MVS Management](#).

Defining APPC/MVS Logon Mode Entry in VTAMLIB

A logon mode is a set of parameters and protocols that determines the communication characteristics of a VTAM session. Logon modes are entries in a logon mode table contained in the system VTAMLIB library.

APPC/MVS requires a logon mode entry in a logon mode table. The logon mode table containing the APPC/MVS logon mode entry must be assembled and linked into the VTAMLIB library defined by the VTAM start procedure. System SAMPLIB member ATBLJOB (for example, SYS1.SAMPLIB(ATBLJOB)) provides sample JCL for assembling and linking a logon mode table.

Figure 15 on page 37 is an example of a logon mode table containing several logon mode entries. This example can be found in system SAMPLIB member GDELOGMD (for example, SYS1.SAMPLIB(GDELOGMD)) or see [“GDELOGMD” on page 51](#).

```

LOGMODES MODETAB  ← VTAM APPL LOGMODE table name
EJECT
*****
      TITLE 'SNASVCMG'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING
*
*      AS LU 6.2 DEVICES
*      REQUIRED FOR LU MANAGEMENT
*****
SNASVCMG MODEENT LOGMODE=SNASVCMG,FMPROF=X'13',TSPROF=X'07',
                PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',
                RUSIZES=X'8585',ENCR=B'0000',
                PSERVIC=X'060200000000000000000300'
*****
      TITLE 'QPCSUPP'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING
*
*      AS LU 6.2 DEVICES
*      REQUIRED FOR LU MANAGEMENT
*****
QPCSUPP MODEENT LOGMODE=QPCSUPP,FMPROF=X'13',TSPROF=X'07',
                PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',
                RUSIZES=X'8585',ENCR=B'0000',
                PSERVIC=X'060200000000000000000300'
*****
      TITLE 'APPCPCLM'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING
*
*      AS LU 6.2 DEVICES
*      FOR PC TARGET
*      IN THIS EXAMPLE THE DEFAULT RU SIZE FOR OS/2 (1024) IS USED
*****
APPCPCLM MODEENT LOGMODE=APPCPCLM,
                RUSIZES=X'8787',
                SRCVPAC=X'00',
                SSNDPAC=X'01'
*****
      TITLE 'APPCHOST'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING
*
*      AS LU 6.2 DEVICES
*      FOR HOST TARGET
*      IN THIS EXAMPLE RU SIZE OF 4096 IS USED
*****
APPCHOST MODEENT LOGMODE=APPCHOST,
                RUSIZES=X'8989',
                SRCVPAC=X'00',
                SSNDPAC=X'01'
MODEEND
END

```

Figure 15. Logon Mode Table

The name of the logon mode table defined in the VTAM APPL definition must match the name defined in VTAMLIB. In [Figure 15 on page 37](#), for example, the name of the logon mode table defined to VTAMLIB is LOGMODES.

For more information about defining the APPC/MVS logon mode, see [z/OS MVS Planning: APPC/MVS Management](#).

Defining LU and Logon Mode on Partner Systems

For DFM to conduct a network conversation with a DDM source implementation, each system in the conversation must know its partner's LU name and logon mode information. Establishing partner information involves the following steps:

1. You need to define on the partner (source) system the local LU, logon mode entry, and partner LU (the LU associated with DFM for z/OS). What conventions and utilities you use for defining this information depends on what platform the partner system runs. On a DDM client system, for example, you use Communications Manager for specifying network information.

The logon mode name on the partner system must match a logon mode table entry name defined to VTAM and associated with the VTAM APPL definition. This association can be explicit in the APPL MODETAB definition statement or implicit in the VTAM-supplied default logon mode table ISTINCLM.

When the partner system sends an APPC allocate call to initiate a conversation with DFM, it sends the name of a logon mode definition that must match a logon mode entry name defined to VTAM. If the partner system is a DDM client, you must use the QPCSUPP logon mode entry name.

2. On the target z/OS system, you need to identify to VTAM the name of the partner LU. This name must match the local LU name that you have established on the partner system.

For information about defining partner information to VTAM, see *VTAM Network Implementation Guide*, SC31-6434; *VTAM Resource Definition Samples*, SC31-6414; and *VTAM Resource Definition Samples*, SC31-6414.

Defining Partner Information on a DDM client

The following is an example of defining a local LU and a VTAM logon mode table specification for a partner a DDM client. This example can be found in system SAMPLIB member GDEPRTL (for example, SYS1.SAMPLIB(GDEPRTL)) or see [“GDEPRTL” on page 54](#).

```
OS2PRTNR LU  LOCADDR=0,
              ISTATUS=ACTIVE,
              MODETAB=LOGMODES ← VTAM Logon Mode Table name
              RESSCB=4
```

Distributed FileManager Customizing Tasks

Customizing DFM includes installing and tuning DFM startup parameters in system PARMLIB, activating the DFM startup procedure in system PROCLIB, and verifying PPT entries for DFM.

Tuning Distributed FileManager Startup Parameters in System PARMLIB

You can tune the startup parameters for DFM to fit your installation's performance requirements. These parameters are contained in PARMLIB member DFM00 (for example, SYS1.PARMLIB(DFM00)). If DFM00 needs to be installed in PARMLIB on your system, you can copy system SAMPLIB member DFM00 (for example, SYS1.SAMPLIB(DFM00)). DFM00 can also be found in [“DFM00” on page 51](#).

The parameters shown in [Table 1 on page 38](#) fall into categories related to either performance tuning or data set definition defaults.

Table 1. Tunable Parameters in DFM00

Parameter	Description	Default	Range
CLOSE_CHECK_INTV	Time interval (in seconds) to wait between searches of the Open PDSE queue for data sets to close	0	0–100
DEFER_CLOSE_TIME	Time interval (in seconds) to wait before closing a PDSE data set after a DDM close command has been processed	0	0–100
LOCK_RETRY	Number of lock conflict retries	3	1–100
LOCK_WAIT_INTV	File lock wait interval in seconds	20	1–100
MAX_AGENT_TSKS	Number of agents supported	5	1–100
MAX_CONV_LOCK	Maximum locks on a file per agent	5	1–100
SEND_BUFFER_THRESHOLD	Maximum number of buffers between APPC SEND verb completions	100	1–1,000
LOGICAL_CACHE	Cache limit for stream files	1,024	1–2,047

Table 1. Tunable Parameters in DFM00 (continued)

Parameter	Description	Default	Range
CCSID	Coded character set identifier (CCSID) for DFM	0	0—65,535
PRIMARY	Data set space allocation in records (non-SMS only)	100	1—2GB
SECONDARY	Data set space allocation in records when PRIMARY space is exhausted (non-SMS only)	50	0—2GB
STREAM_LRECL	Logical record length for stream files	8,196	0—32,760
UNIT	Device type where non-SMS data sets are created, see VOLUME	SYSALLDA	N/A
VOLUME	DASD volume serial number for non-SMS data set creation	None	N/A
RESTRICT_START	Startup command in the PARMLIB DFM00 member	Yes	Yes or No

Parameters Related to Performance

CLOSE_CHECK_INTV and DEFER_CLOSE_TIME

These parameters offer a trade-off between concurrency and PDSE processing performance. If typical usage on your system tends to reaccess the same or other members of a PDSE, these parameters can be specified as nonzero values to leave PDSEs open longer. Then, when PDSE members are reaccessed, the overhead of closing and reopening data sets is eliminated. The trade-off is that the data sets might be unavailable to other remote or local users longer than necessary.

MAX_AGENT_TSKS

This parameter can be used as a control on DFM resources. It determines the maximum number of concurrent remote user tasks that the target server will allow.

MAX_CONV_LOCK

This parameter establishes a limit on how many locks each agent can have. If you think of a lock as representing a system resource (in this case, a data set), then setting a maximum value for the number of locks that can be held establishes a limit on how much serially reusable resource a given agent can use at one time.

LOCK_RETRY and LOCK_WAIT_INTV

These parameters control how soon lock contentions are detected. In an interactive environment where you can choose how to handle “try again later” messages, you might want short wait intervals and few lock conflict retries. However, in a more batch-oriented environment, you might want the opposite to avoid terminating batch jobs just because a lock is temporarily unavailable.

SEND_BUFFER_THRESHOLD

For this parameter, the maximum number of buffers between APPC SEND verb completions should be fairly large to avoid irregularities in system response and to maximize concurrency. Specifying too large a value, however, could result in excess paging.

In some cases, this parameter can increase the overall auxiliary storage requirements of the system. As a general rule, you can determine the auxiliary storage increase by adding up the estimates for the following:

- The total size of the stream-oriented files that are likely to be accessed concurrently by a typical address space
- The space required for input buffers (up to the combined file size)
- The storage required for output buffers (SEND_BUFFER_THRESHOLD times 32k)

Take the resulting sum and multiply it by the number of concurrently running address spaces, then add 25% to allow for control block overhead and unused space at the end of some of the buffers.

LOGICAL_CACHE

This parameter allows you to limit the amount of virtual storage a DFM conversation can use for caching stream files. When the limit is reached, the current stream request is terminated. You can use this parameter to minimize the potential impact of DFM for z/OS on z/OS system performance.

Parameters Related to Data Set Definition

CCSID

Use this parameter to establish the default value for the CCSID associated with data sets that will be created by DFM. This CCSID, unless overridden by the workstation, defines the code page in which stream files are stored when the workstation requests data conversion by specifying the TEXT option. In most cases, the value should be left as zero. Zero is a special value that causes the default CCSID to be inherited from a higher level in the hierarchy. Currently the only value that can be inherited is 500, EBCDIC International Latin-1. The inheritance occurs from DFM itself rather than from the operating system.

If the CCSID is not supported by Character Data Representation Architecture (CDRA), startup will end with message GDE006E indicating that the CCSID keyword has an incorrect value. The return code shown will be that defined for CDRA's CDRGCTL function. If LE is not installed, message GDE006E will be issued for an invalid CCSID with a return code of X'FFFFFFFF'. DFM startup will indicate it is not started, but is actually started in a partial non-data conversion mode.

PRIMARY

This parameter defines the amount of space requested by a user for a data set when it is created. This parameter applied only to the creation of non-SMS data sets. The default primary space allocation is 100 records.

SECONDARY

This parameter defines the amount of additional space requested by the user for a data set when primary space is full. This parameter applied only to the creation of non-SMS data sets. The default secondary space allocation is 50 records.

STREAM_LRECL

This parameter provides a default value for block size if the logical record length is not provided by the SMS DATACLASS. If ACS routines or data classes are established so that all logical record length specifications are provided through SMS data classes, this parameter can be specified as 0. Larger values than 8196 can give better performance, but generally the most important consideration is whether the data will be shared with z/OS applications.

If the data is shared, the needs of the z/OS applications should determine the logical record length. For example, choosing a small value might allow easy editing or browsing of the file on z/OS. If, however, the data will not be shared with z/OS applications, the larger the logical record length the better the performance will tend to be.

UNIT

This parameter defines the device type where non-SMS data sets are created, see VOLUME.

VOLUME

This parameter defines the DASD serial number for non-SMS data set creation.

When DFM is installed, the VOLUME parameter must be activated if the installation does not use SMS or chooses not to establish ACS routines for the DFM.

If SMS is not active and VOLUME is omitted, it is possible to create SAM data sets, but PDS data sets created will not have directory blocks assigned to them and a "file damaged" error will occur.

Parameters Related to DataAgent

RESTRICT_START

This parameter is the startup command in the PARMLIB DFM00 member. The default is YES.

Activating Distributed FileManager in System PROCLIB

Activating DFM involves adding a startup procedure to a new system PROCLIB member called DFM, for example, SYS1.PROCLIB(DFM). Once DFM is added to PROCLIB, DFM can be started by a system operator command. The operator command, which can be part of initial program load, identifies the DFM member (see “Starting Up Distributed FileManager” on page 45).

APPC/MVS and the APPC/MVS transaction scheduler (both are started by operator command) must be active before DFM LU 6.2 conversations can take place on z/OS.

Figure 16 on page 41 is an example of the contents of the DFM member.

```
//DFM      PROC  PARMS='NORMAL '
//*****
//*
//* DFM START UP PROCEDURE
//*
//*****
//DFM      EXEC  PGM=GDEISBOT,
//          PARM='&PARMS',
//          REGION=0K,
//          TIME=1440
//IEFPARM  DD   DSN=SYS1.PARMLIB,DISP=SHR
//* CHANGE THE STEPLIB STATEMENT AS REQUIRED IF YOUR INSTALLATION
//* DOESN'T HAVE THE LE RUNTIME DATA SET IN ITS LINK LIST.
//*STEPLIB DD   DSN=SYS1.SCEERUN,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//*
//* THE TWO FILES ASSOCIATED WITH THE DD STATEMENTS CDRATRC AND
//* SYSOUT CAN BE USED TO DIAGNOSE DFM STARTUP PROBLEMS RELATED
//* TO CDRA. (CDRA IS INVOKED DURING STARTUP FOR CERTAIN CCSID
//* VALUES IN THE SYS1.PARMLIB MEMBER DFM00.)
//*
//* YOU MUST ALLOCATE THE TWO FILES AS RECFM=FBA, LRECL=133,
//* AND DSORG=PS BEFORE STARTING DFM WITH THE DD STATEMENTS
//* ACTIVE.
//*
//* NOTE THAT SYSOUT IS REQUIRED AND CDRATRC IS OPTIONAL
//* WHEN USING CDRA AND THE DEFAULT INSTALLATION IS SET UP TO
//* USE CDRA IF YOUR HOST CODE PAGE IS OTHER THAN 500.
//*
//* CDRATRC DD   DSN=SYS1.CDRATRC,DISP=SHR      CDRA API TRACING
//* SYSOUT  DD   DSN=SYS1.CDRAOUT,DISP=SHR      C RUNTIME MESSAGES
//SYSOUT   DD   DUMMY <- DEFAULT = CDRA WITH RUNTIME MESSAGES DISCARDED
```

Figure 16. DFM Member Example

Note: LE is required to use CDRA, if LE is installed and is not in the link list, SYS1.PROCLIB(DFM) and SYS1.SAMPLIB(GDETPDEF) should be modified so their STEPLIB DD statements refer to the proper LE run time library. Refer to DFMREADM in SYS1.SAMPLIB for details.

Verifying PPT Entries for Distributed FileManager

To execute correctly, DFM must have entries in the system program property table (PPT). These entries are automatically included in the base PPT for your installation (system LINKLIB member IEFSDPPT). If the need arises to override this base PPT, you can add the entries to system PARMLIB member SCHEDxx (for example, SYS1.PARMLIB(SCHEDxx)). For a sample of the entries, see Chapter 6, “PPT Entries for Distributed FileManager,” on page 85. PARMLIB(SCHEDxx) members for these sample entries should not be created without prior discussion with your IBM service representative.

ACS Routines for Defining Distributed FileManager SMS Classes

ACS routines determine the SMS classes for data sets. For data sets to be classified as SMS-managed, they must be defined in a storage class. DFM only permits remote creation of data sets when the resultant data set is SMS-managed. If a request to create a data set would result in a non-SMS-managed data set, DFM rejects the request.

DFM does not support the use of large format data sets, which are physical sequential data sets with the ability to grow beyond the previous size limit of 65 535 tracks per volume. To prevent attempts to create a large format data set, it is recommended that provide a storage class ACS routine to fail any DFM request to create an unsupported type, such as &DSNTYPE = EXR, EXC, or LARGE.

For more information about large format data sets, see SC26-7410z/OS DFSMS Using Data Sets.

Figure 17 on page 42, Figure 18 on page 42, and Figure 19 on page 42 are sample ACS routines for defining data, management, and storage classes for data sets created by DFM.

```

/*          DATACLAS ROUTINE                                */
/* DEFAULT DATACLASSES FOR DFM                             */
/*                                                                 */
IF &JOB = 'GDEDFM' AND &DATACLAS = '' THEN
DO
  SELECT
    WHEN (&RECORG = 'KS') SET &DATACLAS = 'KS000000'
    WHEN (&RECORG = 'ES') SET &DATACLAS = 'ES000000'
    WHEN (&RECORG = 'RR') SET &DATACLAS = 'RR000000'
    WHEN (&DSORG = 'PS') SET &DATACLAS = 'PS000000'
    WHEN (&DSNTYPE = 'LIBRARY') SET &DATACLAS = 'LIB000000'
    OTHERWISE WRITE 'NOT A SUPPORTED DFM DATASET TYPE'
  END /* SELECT */
  /* DEBUGGING STATEMENT FOLLOWS. REMOVE IT WHEN ROUTINE IS OK. */
  IF &DATACLAS ^= '' THEN
    WRITE 'DATACLAS SET TO '&DATACLAS' FOR DFM'
  EXIT CODE(0)
END /* DO */

```

Figure 17. Data Class Routine

If the logical record length for stream files is specified as zero (STREAM_LRECL(0)) in the system PARMLIB member DFM00, you must select a data class providing a nonzero record length.

```

/*          MGMTCLAS ROUTINE                                */
/* IF JOB IS DFM                                             */
/* SET MGMTCLAS TO DFMMGMT.                                  */
/*                                                                 */
IF &JOB = 'GDEDFM' THEN
DO
  SET &MGMTCLAS = 'DFMMGMT'
  /* DEBUGGING STATEMENT FOLLOWS. REMOVE IT WHEN ROUTINE IS OK. */
  WRITE '&MGMTCLAS SET TO '&MGMTCLAS' FOR DFM'
  EXIT CODE(0)
END /* GDEDFM */
/*                                                                 */

```

Figure 18. Management Class Routine

```

/*          STORCLAS ROUTINE                                */
/* DEFAULT STORCLAS FOR DFM IS DFMCLASS.                    */
/*                                                                 */
/*                                                                 */
IF &JOB = 'GDEDFM' AND &STORCLAS = '' THEN
DO
  SET &STORCLAS = 'DFMCLASS'
  /* DEBUGGING STATEMENT FOLLOWS. REMOVE IT WHEN ROUTINE IS OK. */
  WRITE 'STORCLAS SET TO '&STORCLAS' FOR DFM'
  EXIT CODE(0)
END
/*                                                                 */
/*                                                                 */

```

Figure 19. Storage Class Routine

Data sets without a storage class cannot be SMS-managed.

Establishing Distributed FileManager TP Access Security

You need to establish access security for the DFM TP so that only authorized users and applications can remotely access it. To protect the DFM TP, you can:

- Limit which LUs can enter your z/OS system
- Ensure that inbound requests to initiate conversations with the DFM TP contain security information such as user IDs and passwords
- Limit by user ID or group who can access the DFM TP
- Limit the administrators who can define and update information in the DFM TP profile

Using RACF to Control Access to the Distributed FileManager TP

You can use RACF (or an equivalent product) to control which user IDs or groups of user IDs are authorized to access DFM. To accomplish this, you need the following information:

- Name of your DFM TP profile
- User IDs that will be authorized EXECUTE access to your APPC/MVS TP
- User IDs that will be authorized as APPC/MVS administrators to read and update DFM TP profile information

The RACF APPCTP resource class controls the use of the APPC/MVS TP. Profiles in this resource class define which user IDs can execute the APPC/MVS TP. The names of these profiles are in the form *dbtoken.level.tpname*, where

dbtoken

The database token associated with the DFM TP profile (1 to 8 characters). The TP profile must have a database token, or else APPC/MVS cannot call RACF for TP access security. For more information about adding a database token, see [z/OS MVS Planning: APPC/MVS Management](#).

level

This is one of the following:

- The name of your system library (for example, SYS1), if the TP is available to all users who can access the LU
- A group ID, if the TP is available to a group
- A user ID, if the TP is available to just a specific user

tpname

The name of the DFM TP profile, which is always ^X'07'001 (see [“Adding the TP Profile to the VSAM KSDS”](#) on page 33)

Defining the Distributed FileManager TP Profile to RACF

The following example defines to RACF the DFM TP profile name (^X'07'001) in the RACF APPCTP class:

```
RDEFINE APPCTP TOKEN1.SYSTEM.^X'07'001 UACC(NONE)
```

Defining a TP Administrator to RACF

The following example defines to RACF the user ID ADMIN01 with update access to the DFM TP profile:

```
PERMIT TOKEN1.SYSTEM.^X'07'001 CLASS (APPCTP) ID (ADMIN01) ACCESS(UPDATE)
```

Defining a User ID to RACF

The following example defines to RACF the user ID DFMUSER with authorization to execute the DFM TP:

```
PERMIT TOKEN1.SYSTEM.^X'07'001 CLASS (APPCTP) ID (DFMUSER) ACCESS(EXECUTE)
```

Implementing RACF Access Protection for TP

To implement RACF protection as defined in the APPCTP profile, you must activate in RACF the APPCTP class and SETROPTS RACLIST for the class. For example:

```
SECTROPTS CLASSACT(APPCTP) RACLIST(APPCTP)
```

For more detailed information about using RACF to control DFM TP access, see [z/OS MVS Planning: APPC/MVS Management](#).

Chapter 4. Operating Distributed FileManager

This chapter is about operating Distributed FileManager (DFM) on a z/OS system. It covers procedures for starting up the DFM environment on z/OS and for monitoring and controlling the status of DFM conversations.

For more information, see *z/OS MVS Planning: APPC/MVS Management*.

Starting the Distributed FileManager Environment

Starting the DFM environment requires that computer operations run procedures to start APPC/MVS, the APPC/MVS transaction scheduler, and DFM.

APPC/MVS and the transaction scheduler must be started before starting DFM. To automate these procedures at initial program load (IPL), you can add the startup commands to the system PARMLIB member COMMNDxx (for example, SYS1.PARMLIB(COMMNDxx)).

Starting APPC/MVS

The startup parameters for APPC/MVS are in system PARMLIB member APPCPMxx. These parameters define the local LU to be used for APPC/MVS. They associate the LU with an APPC/MVS transaction scheduler and the DFM TP profile. See [“Defining PARMLIB Start Parameters for APPC/MVS” on page 32](#) for more details.

The following command starts up APPC/MVS:

```
START APPC,SUB=MSTR,APPC=xx
where xx is the unique APPCPMxx suffix
```

Starting the APPC/MVS Transaction Scheduler

The APPC/MVS transaction scheduler initiates and schedules TPs in response to inbound allocate requests. The system PARMLIB member ASCHPMxx controls the transaction scheduler for the DFM TP. For more information about creating ASCHPMxx, see [“Defining PARMLIB Start Parameters for the APPC/MVS Scheduler” on page 34](#).

The following command starts up the APPC/MVS transaction scheduler:

```
START ASCH,SUB=MSTR,ASCH=xx
where xx is the unique ASCHPMxx suffix
```

Starting Up Distributed FileManager

System PROCLIB member DFM contains the procedure for starting up &mv sdfmtemp (see [“Activating Distributed FileManager in System PROCLIB” on page 41](#)). DFM must be active prior to APPC/MVS initiating a conversation.

The following command starts DFM:

```
START DFM,SUB=MSTR
```

Triggering the Distributed FileManager DataAgent

A DFM for z/OS DataAgent can only be triggered from an SdU application or a DDM application that is written to call the DataAgent from a client workstation. IBM provides sample DataAgent routines, DFMXAGNT, DFMQTSO, DFMXSORT, and DFMXTSO that you can execute. You can also use these routines as examples to help write your own DataAgent routines. Your SdU application or DDM application uses

the DDMOpen function to trigger the DataAgent processing on z/OS and it uses the DDMClose function to terminate the DataAgent processing.

Monitoring Status of Distributed FileManager Conversations

APPC/MVS provides the DISPLAY command for monitoring the status of APPC/MVS conversations. The DISPLAY APPC command gives status information about TPs and LUs. The DISPLAY ASCH command gives status information about APPC/MVS transaction schedulers. [“Using the DISPLAY APPC Command” on page 46](#) provides examples of using these commands.

Using the DISPLAY APPC Command

These are examples of using the DISPLAY APPC command.

Displaying TP Status Information

These examples use the DISPLAY APPC command to return selected TP status information about the following:

- DFM TP:

```
DISPLAY APPC,TP,LIST,LTPN=^X'07'001
```

- TPs scheduled by ASCH:

```
DISPLAY APPC,LIST,SCHED=ASCH
```

- DFM TP in a particular address space:

```
DISPLAY APPC,LIST,ASID=asid  
where asid is the hexadecimal address space identifier
```

- TPs activated by a specific user ID:

```
DISPLAY APPC,LIST,USERID=userid
```

Displaying LU Status Information

These examples use the DISPLAY APPC command to return selected LU status information about the following:

- A local LU:

```
DISPLAY APPC,LU,LIST,LLUN=lluname  
where lluname is the name of a local LU
```

- All LUs (includes detailed information about local and partner LUs):

```
DISPLAY APPC,LU,ALL
```

Using the DISPLAY ASCH Command

These examples use the DISPLAY ASCH command to return selected status information about ASCH:

- Summary of ASCH transaction scheduling information (includes summary of all APPC/MVS transaction scheduling activity):

```
DISPLAY ASCH,SUMMARY
```

- DFM TP scheduling information:


```
DISPLAY ASCH,LIST,ASID=001E
where 001E is the hexadecimal address space identifier for
the Distributed FileManager TP
```

- TPs scheduled by a specific user ID:

```
DISPLAY ASCH,LIST,USERID=userid
```

Controlling Status of Distributed FileManager Conversations

This discussion covers the following ways for controlling the status of DFM conversations:

- Deactivating the DFM TP
- Stopping a local LU with the SET command
- Stopping DFM with the CANCEL command
- Using the FORCE command

Deactivating the Distributed FileManager TP

You can deactivate the DFM TP by modifying its TP profile. Using the TP MODIFY command in the APPC/MVS administration utility (ATBSDFMU), you can stop the DFM TP from being scheduled and stop new requests for the TP.

The TP MODIFY command lets you change the active status of the DFM TP to NO in the TP profile data set. If the DFM TP is running at the time, then the current and any queued requests are allowed to complete. No new requests, however, are allowed.

This is an example of using the TP MODIFY command to deactivate the DFM TP:

```
TPMODIFY
TPNAME(^X'07'001)
SYSTEM
ACTIVE(NO)
```

For more information, see [z/OS MVS Planning: APPC/MVS Management](#).

Stopping a Local LU with the MVS SET Command

You can stop work from coming into a local LU by using the MVS™ SET command to delete an LU from the APPC/MVS configuration. Use this method to:

- Stop an LU that is not functioning properly (for example, because of a VTAM error)
- Stop TPs defined in a TP profile data set that uses one or more LUs
- Stop a TP scheduler

To stop an LU by using the SET command:

1. Code system PARMLIB member APPCPMxx with the command to delete the LU. For example, to delete an LU named MYLU, code PARMLIB member APPCPM1D as follows:

```
LUDEL
ACBNAME(MYLU)
```

2. After coding APPCPM1D, issue this command to stop the LU:

```
SET APPC=1D
```

For more information about deleting LUs, see [z/OS MVS Planning: APPC/MVS Management](#).

Stopping DFM for z/OS with the MVS CANCEL Command

You can use the MVS CANCEL command to immediately stop the DFM TP in a particular address space. It can also be used to immediately stop the DFM startup procedures APPC/MVS, ASCH, and DFM. If the CANCEL command is not successful, try the FORCE command (see [“Using the FORCE Command” on page 48](#)).

Stopping the Distributed FileManager TP

This example stops both the DFM TP and any associated APPC/MVS conversation:

1. First use this command to find out the jobname and address space identifier (ASID) for your DFM TP:

```
DISPLAY ASCH,ALL,LTPN=^X'07'001
where ^X'07'001 is the Distributed FileManager TP profile name
```

2. Suppose that the jobname is GDEDFM and the ASID is 0044. Use this information as shown to immediately stop the DFM TP:

```
CANCEL GDEDFM,A=0044
```

Stopping APPC/MVS, ASCH, and DFM

The following examples stop APPC/MVS, ASCH, and DFM:

```
CANCEL APPC
CANCEL ASCH
CANCEL DFM
```

You should consider the following before using the CANCEL command:

- Before cancelling DFM, first cancel all jobs servicing APPC/MVS conversations, for example JOBNAME GDEDFM. Otherwise, the jobs abend when they try to access DFM resources.
- Each time DFM is cancelled, the system marks the address space in which it was running as nonreusable until the next IPL.
- Cancelling APPC, ASCH, or DFM immediately ends all TPs and scheduling activity for APPC/MVS, which could have serious repercussions.

Note: 13E abends can occur during CANCEL command processing. These abends are perfectly normal and do not interfere with the CANCEL command processing.

Using the FORCE Command

If the CANCEL command is not successful, you can try the MVS FORCE command to stop the DFM TP. Using the FORCE command can, however, result in loss of resources until the system is re-IPLed.

In this example, the FORCE command is used to stop a DFM TP with jobname GDEDFM and an ASID of 0044:

```
FORCE GDEDFM,A=0044
```

Chapter 5. System Samples

This topic documents:

- The system SAMPLIB and PROCLIB samples related to customizing z/OS for the Distributed FileManager environment
- A sample of the PPT entries for Distributed FileManager
- The DFM DataAgent DFMACALL.C sample

System SAMPLIB Samples

The following are system SAMPLIB samples that are referred to in [Chapter 3, “Customizing z/OS for Distributed FileManager,”](#) on page 29.

GDEAPPC

System SAMPLIB member GDEAPPC, for example, SYS1.SAMPLIB(GDEAPPC), contains the sample shown in [Figure 20 on page 49](#) of the APPC/MVS start parameters.

```
/* START OF SPECIFICATIONS *****/
/*
/*01* MEMBER-NAME: GDEAPPC
/*
/*02* DESCRIPTIVE-NAME: DFSMS DISTRIBUTED FILEMANAGER SAMPLE TO
/*                      DEFINE A LOCAL LU TO APPC/MVS
/*
/*01* DISCLAIMER =
/*
/* THIS SAMPLE SOURCE IS PROVIDED FOR TUTORIAL PURPOSES ONLY. A
/* COMPLETE HANDLING OF ERROR CONDITIONS HAS NOT BEEN SHOWN OR
/* ATTEMPTED, AND THIS SOURCE HAS NOT BEEN SUBMITTED TO FORMAL IBM
/* TESTING. THIS SOURCE IS DISTRIBUTED ON AN 'AS IS' BASIS
/* WITHOUT ANY WARRANTIES EITHER EXPRESSED OR IMPLIED.
/*
/*01* FUNCTION:
/* THIS SAMPLE MEMBER DEFINES AN LU TO APPC, ALONG WITH A VSAM
/* DATASET FOR TP PROFILES AND A SECOND ONE FOR SIDE INFORMATION
/*
/*01* DISTRIBUTION LIBRARY: ASAMPLIB
/*
/*01* CHANGE-ACTIVITY:
/*
/* FLAG LINEITEM FMID DATE ID COMMENT
/* $L0=GDEAPPC HDZ11B0 931009 DFSMS 1.2.0 DISTRIBUTED FILEMANAGER
/*                      SAMPLE TO ADD A LOCAL LU TO APPC/MVS
/*
/*
/*
/*
/*****
LUADD ACBNAME(MVSLU01) BASE TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
/*****/
```

Figure 20. APPC/MVS Start Parameters

GDEAPDEF

System SAMPLIB member GDEAPDEF, for example, SYS1.SAMPLIB(GDEAPDEF), contains the sample shown in [Figure 21 on page 50](#) of a VTAM APPL definition in VTAMLST.

```

*/ * START OF SPECIFICATIONS *****
*
*01* MEMBER-NAME: GDEAPDEF
*
*02* DESCRIPTIVE-NAME: SAMPLE VTAM APPL STATEMENT FOR APPC/MVS
*                      NECESSARY TO RUN DFSMS DISTRIBUTED FILEMANAGER
*
* 01* DISCLAIMER =
*
*   THIS SAMPLE SOURCE IS PROVIDED FOR TUTORIAL PURPOSES ONLY. A
*   COMPLETE HANDLING OF ERROR CONDITIONS HAS NOT BEEN SHOWN OR
*   ATTEMPTED, AND THIS SOURCE HAS NOT BEEN SUBMITTED TO FORMAL IBM
*   TESTING. THIS SOURCE IS DISTRIBUTED ON AN 'AS IS' BASIS
*   WITHOUT ANY WARRANTIES EITHER EXPRESSED OR IMPLIED.
*
*
*01* FUNCTION: THIS APPL STATEMENT IDENTIFIES APPC/MVS AS A VTAM
*              APPLICATION, WITH ONE ACB DEFINED FOR LU MVSLU01.
*
*
*01* DISTRIBUTION LIBRARY: ASAMPLIB
*
*01* CHANGE-ACTIVITY:
*
* FLAG LINEITEM FMID   DATE ID COMMENT
* $L0=GDEAPDEF HDZ11B0 931009 DFSMS 1.2.0 DISTRIBUTED FILEMANAGER
*                      SAMPLE VTAM APPL DEFINITION
*
*****
MVSLU01 APPL  ACBNAME=MVSLU01,
              APPC=YES,
              AUTOSES=0,
              DDRAINL=NALLOW,
              DMINWNL=5,
              DMINWNR=5,
              DRESPL=NALLOW,
              DSESLIM=10,
              LMDENT=19,
              MODETAB=LOGMODES,
              PARSESS=YES,
              SECACPT=CONV,
              SRBEXIT=YES,
              VPACING=1

```

Figure 21. VTAM APPL Definition in VTAMLST

GDEASCH

System SAMPLIB member GDEASCH, for example, SYS1.SAMPLIB(GDEASCH), contains the sample shown in [Figure 22 on page 51](#) of start parameters for the APPC/MVS scheduler (ASCH).

```

/** START OF SPECIFICATIONS *****/
/*
/*01*MEMBER-NAME: GDEASCH
/*
/*02* DESCRIPTIVE-NAME: SAMPLE ASCH START PARAMETER STATEMENTS
/*
/*          NECESSARY TO RUN DFSMS DISTRIBUTED FILEMANAGER*/
/*
/*01* DISCLAIMER =
/*
/* THIS SAMPLE SOURCE IS PROVIDED FOR TUTORIAL PURPOSES ONLY. A
/* COMPLETE HANDLING OF ERROR CONDITIONS HAS NOT BEEN SHOWN OR
/* ATTEMPTED, AND THIS SOURCE HAS NOT BEEN SUBMITTED TO FORMAL IBM
/* TESTING. THIS SOURCE IS DISTRIBUTED ON AN 'AS IS' BASIS
/* WITHOUT ANY WARRANTIES EITHER EXPRESSED OR IMPLIED.
/*
/*01* FUNCTION:
/* THIS PARMLIB MEMBER SETS UP A SCHEDULER CLASS.
/*
/*01* DISTRIBUTION LIBRARY: ASAMPLIB
/*
/*01* CHANGE-ACTIVITY:
/*
/*FLAG LINEITEM  FMID  DATE  ID  COMMENT
/* $L0=GDEASCH  HDZ11B0 931009 DFSMS 1.2.0 DISTRIBUTED FILEMANAGER
/*          SAMPLE ASCH START PARAMETERS
/*
/*
/*****
CLASSADD CLASSNAME(A)
        MSGLIMIT(1000) MAX(10) MIN(1) RESPGOAL(1)

```

Figure 22. ASCH Start Parameter Statements to Run DFM

DFM00

System SAMPLIB member DFM00, for example, SYS1.SAMPLIB(DFM00), contains the sample shown in [Figure 23 on page 51](#) of the startup parameters for Distributed FileManager.

```

DFM CCSID(0)
  CLOSE_CHECK_INTV(0)
  DEFER_CLOSE_TIME(0)
  LOCK_RETRY(3)
  LOCK_WAIT_INTV(20)
  LOGICAL_CACHE(1024)
  MAX_AGENT_TSKS(5)
  MAX_CONV_LOCK(5)
  RESTRICT_START(YES)
  STREAM_LRECL(8196)
  SEND_BUFFER_THRESHOLD(100)
/* Uncomment the next lines to provide allocation defaults if
/* non-SMS files are to be created by DFM. Remove or comment them
/* out again once SMS allocation is in use.
/* PRIMARY(100) SECONDARY(50) */
/* UNIT(SYSALLDA)
/* VOLUME(xxxxxx)

```

Figure 23. Startup Parameters for Distributed FileManager

GDELOGMD

System SAMPLIB member GDELOGMD, for example, SYS1.SAMPLIB(GDELOGMD), contains the sample shown in [Figure 24 on page 52](#) of a VTAM logon mode table that contains the logon mode entry for Distributed FileManager.


```

      TITLE 'QPCSUPP'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING      *
*      AS LU 6.2 DEVICES                                           *
*      REQUIRED FOR LU MANAGEMENT                                   *
*****
QPCSUPP  MODEENT LOGMODE=QPCSUPP,FMPROF=X'13',TSPROF=X'07',
          PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',
          RUSIZES=X'8585',ENCR=B'0000',
          PSERVIC=X'0602000000000000000000000300'
*****
      TITLE 'APPCPCLM'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING      *
*      AS LU 6.2 DEVICES                                           *
*      FOR PC TARGET                                              *
*      IN THIS EXAMPLE THE DEFAULT RU SIZE FOR OS/2 (1024) IS USED *
*****
APPCPCLM MODEENT LOGMODE=APPCPCLM,
          RUSIZES=X'8787',
          SRCVPAC=X'00',
          SSNDPAC=X'01'
*****
      TITLE 'APPCHOST'
*****
*      LOGMODE TABLE ENTRY FOR RESOURCES CAPABLE OF ACTING      *
*      AS LU 6.2 DEVICES                                           *
*      FOR HOST TARGET                                            *
*      IN THIS EXAMPLE RU SIZE OF 4096 IS USED                    *
*****
APPCHOST MODEENT LOGMODE=APPCHOST,
          RUSIZES=X'8989',
          SRCVPAC=X'00',
          SSNDPAC=X'01'
MODEEND
END

```

Figure 25. VTAM Logon Mode Table Part 2 of 2

GDETPDEF

System SAMPLIB member GDETPDEF, for example, SYS1.SAMPLIB(GDETPDEF), contains the sample shown in [Figure 26 on page 54](#) of adding the Distributed FileManager TP profile to the APPC/MVS TP profile data set.

Note: The GDEDFM job in GDETPDEF should have either no region size or a region size of 0K to contain cached stream files. LE is required to use CDRA, if LE is installed and is not in the link list, SYS1.PROCLIB(DFM) and SYS1.SAMPLIB(GDETPDEF) should be modified so their STEPLIB DD statements refer to the proper LE run time library. Refer to DFMREADM in SYS1.SAMPLIB for details. SYSOUT and CDRATRC files can be allocated as RECFM=FBA, LRECL=133, and DSORG=PS for use in diagnosing CDRA problems.

```

//*****
//* PROPRIETARY V2 STATEMENT
//* LICENSED MATERIALS - PROPERTY OF IBM
//* 5695-DF1 (C) COPYRIGHT 1994,1995 IBM CORP.
//* END PROPRIETARY V2 STATEMENT
//*
//*****
//IBMUSER1 JOB 'GDETPDEF',NOTIFY=IBMUSER,MSGCLASS=H
//*****
//*
//* GDETPDEF - MVS/APPC setup for DFM: TP definition utility
//*
//* This job invokes the APPC/MVS administration utility to add
//* the TP profile to the APPC/MVS data set.
//*
//* It consists of a single job step that adds a MVS/DFM TP
//* to SYS1.APPCTP.
//*
//* Modify the above job statement as required and,
//* optionally, make the following modifications
//* to the job itself:
//*
//*   change 'SYS1.APPCTP' to another name if required by
//*       your installation
//*   change the DFMJOB card to one suitable for your installation
//* Note that you can alter the DD statements CDRATRC and SYSOUT
//* as needed to obtain CDRA API trace output and C runtime messages.
//*****

```

Figure 26. MVS/APPC Setup for DFM: TP Definition Utility Part 1 of 2

```

//STEP      EXEC  PGM=ATBSDFMU
//SYSPRINT  DD    SYSOUT=*
//SYSSDOUT  DD    SYSOUT=*
//SYSSDLIB  DD    DSN=SYS1.APPCTP,DISP=SHR
//SYSIN     DD    DATA,DLM=XX
TPDELETE
TPNAME(^X'07'001)
TPADD
TPNAME(^X'07'001)
ACTIVE(YES)
TPSCHED_DELIMITER(##)
CLASS(A)
JCL_DELIMITER(ENDJCL)
//GDEDFM JOB MSGCLASS=H,MSGLEVEL=(1,1),CLASS=A
//GDEDFM EXEC PGM=GDEISASB
//* CHANGE THE STEPLIB STATEMENT AS REQUIRED IF YOUR INSTALLATION
//* DOES NOT HAVE THE LE RUNTIME DATA SET IN ITS LINK LIST.
//*STEPLIB DD DSN=SYS1.SCEERUN,DISP=SHR
//*CDRATRC DD DSN=SYS1.CDRATRC2,DISP=SHR <- CDRA API TRACE OUTPUT
//*SYSOUT DD DSN=SYS1.CDRAOUT2,DISP=SHR <- C RUNTIME MESSAGES
//SYSOUT DD DUMMY <- C RUNTIME MESSAGES (NO-OP)
ENDJCL
##
XX

```

Figure 27. MVS/APPC Setup for DFM: TP Definition Utility Part 2 of 2

GDEPRTL

System SAMPLIB member GDEPRTL, for example, SYS1.SAMPLIB(GDEPRTL), contains the sample shown in [Figure 28 on page 55](#) of a partner definition for a DDM client system.


```

*/ * START OF SPECIFICATIONS *****
*
*01* MEMBER-NAME: GDEPRTL U
*
*02* DESCRIPTIVE-NAME: SAMPLE VTAM PARTNER LU DEFINITION NECESSARY
*                      TO RUN DFSMS DISTRIBUTED FILEMANAGER
*
* 01* DISCLAIMER =
*
*   THIS SAMPLE SOURCE IS PROVIDED FOR TUTORIAL PURPOSES ONLY. A
*   COMPLETE HANDLING OF ERROR CONDITIONS HAS NOT BEEN SHOWN OR
*   ATTEMPTED, AND THIS SOURCE HAS NOT BEEN SUBMITTED TO FORMAL IBM
*   TESTING. THIS SOURCE IS DISTRIBUTED ON AN 'AS IS' BASIS
*   WITHOUT ANY WARRANTIES EITHER EXPRESSED OR IMPLIED.
*
*01* FUNCTION: THIS LU STATEMENT IDENTIFIES THE PARTNER LU
*
*01* DISTRIBUTION LIBRARY: ASAMPLIB
*
*01* CHANGE-ACTIVITY:
*
* FLAG LINEITEM FMID DATE ID COMMENT
* $L0=GDEPRTL U HDZ11B0 931009 DFSMS 1.2.0 DISTRIBUTED FILEMANAGER
*                      SAMPLE PARTNER LU DEFINITION
*
*****
OS2PRTNR LU LOCADDR=0,
           ISTATUS=ACTIVE,
           MODETAB=LOGMODES,
           RESSCB=4

```

Figure 28. VTAM Partner LU Definition to Run DFSMS/DFM

DFMX0001

System SAMPLIB member DFMX0001, for example, SYS1.SAMPLIB(DFMX0001), contains the sample shown in [Figure 29 on page 56](#), showing how to set up a procedure for starting the DFM DataAgent.

```

//DFMX0001 JOB ,MSGCLASS=Z
//DFMX0001 PROC DFMINIT= <,optional_procedural_parameters>
//DFMAGENT EXEC PGM=&DFMINIT <,optional_program_parameters>
//*
/* This procedure is a sample showing how to set up a procedure
/* for starting a DFM DataAgent.
/*
/* DFM DataAgent processing requires a procedure (whose name is
/* the same as the agent name). The procedure has to run DFMINIT
/* and DFMINIT will call a DataAgent routine. The DataAgent routine
/* will default to the same name as the agent (or procedure) name
/* but can be the name of any executable program suitable for
/* running as a key 8 job step.
/*
/* For example, you could run this DataAgent routine with no
/* further setup by issuing the following SdU sample command
/* from a workstation:
/*      dfmacall agent x:filename dfmx0001 pgm iefbr14
/*
/* DFMINIT is DFM's DataAgent routing module and should not
/* be changed. Other DD statements and symbolic
/* substitutions can be added as needed by the DataAgent
/* program itself.
/*
/* This example is intended to discard the output by routing
/* it to MSGCLASS of Z. You should modify it as appropriate
/* for your installation. If you do not purge output, the
/* operator will have to periodically have to issue the $ps
/* command.
/*
/* Add additional DD statements as required by your program.
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DFMX0001 PEND
//GO      EXEC DFMX0001

```

Figure 29. Starting the DFM DataAgent

DFMXAGNT

System SAMPLIB member DFMXAGNT, for example, SYS1.SAMPLIB(DFMXAGNT) shown in [Figure 30 on page 57](#), contains the DFM DataAgent sample routine.

```

/*****
* PROPRIETARY V3 STATEMENT
* LICENSED MATERIALS - PROPERTY OF IBM
* 5695-DF1
* (C) COPYRIGHT 1997 IBM CORP.
* END PROPRIETARY V3 STATEMENT
*****/

/*****
*
* $MOD(DFMXAGNT) COMP(5695-DF120)
*
* MODULE NAME: DFMXAGNT
*
* DESCRIPTION: DFM DataAgent Sample Routine
*
* STATUS: Version 1 Release 4.0 (DFSMS)
*
* COPYRIGHT: See the copyright statement on the previous page.*
*
* FUNCTION: This module illustrates how a DFM DataAgent can
* be written in C. It gets control at file declaration time
* and sets the reason code so that it will also get control
* when the file declaration is deleted. Its sole function is
* to issue printf statements displaying the parameters if its
* internal debug flag is set.
*
* You can use the C compiler and your installation's linkedit
* JCL to build the DFMXAGNT executable code on the mainframe.
*
* You can then copy DFMX0001 to produce a proclib
* member (proclib member name is also referred to as the
* DFM DataAgent name). You could then invoke the DataAgent
* by means of the DFMACALL sample application provided by
* SdU. (An analogous procedure is given by the installation
* sample DFMXSRTI but in this case you might call the load
* module DFMXAGNT and copy DFMX0001 to build PROCLIB(DFMXAGNT)
* so that "dfmacall agent x:filename dfmxagent" could be used
* to invoke it from the workstation.)
*
* An installation sample is not provided in order to
* demonstrate that the manual process as outlined above is
* straight-forward and because this sample would probably not
* be used without significant changes anyway.
*
*****/

```

Figure 30. DFM DataAgent Sample Routine Part 1 of 5

```

* PROCESSING: *
*
* LOGIC: *
* Refer to block comments in the code. *
*
* ERROR PROCESSING: *
* Issue a printf and then return with register 15 set to a *
* non-zero value and with the reason code in the extended *
* parameter list set to a unique value. *
*
* NOTES: *
*
* PATCH SPACE: None *
* XAX CONSIDERATIONS: AMODE(31) RMODE(ANY) ENV(PRI) *
* DEPENDENCIES: The Language Environment and the C runtime *
* library must be installed. *
* RESTRICTIONS: None *
* REGISTER CONVENTIONS: Standard conventions--refer to C *
* compiler documentation. *
* SERIALIZATION: No serialization techniques are used by this *
* module. *
*
* MODULE TYPE: Procedure *
* PROCESSOR: C *
* ATTRIBUTES: *
* TYPE: Reentrant *
* PRIMARY ASID: Caller's ASID *
* SECONDARY ASID: Same as primary *
* HOME ASID: Same as primary *
* MODE: Task *
* KEY: 8 *
* STATE: Problem program *
* LOCATION: Link library *
*
* ENTRY POINT: main *
*
* PURPOSE: Show that a DataAgent routine can be written in C. *
* LINKAGE: Called by Distributed FileManager. *
* INPUT/OUTPUT: Refer to the DFM Guide and Reference for the *
* parameter list format. *
*
* MESSAGES: Refer to printf statements. *
*
* EXIT NORMAL: *
* RETURN CODE: Register 15 = 0 *
* REASON CODE: Not applicable *
* MESSAGE ID: None: *

```

Figure 31. DFM DataAgent Sample Routine Part 2 of 5

```

*
* EXIT ERRORS:
* RETURN CODE: Register 15 = 8
* REASON CODE: Unique values set in the extended parameter
* list
* MESSAGE ID: See printf statements.
*
* EXTERNAL REFERENCES: None
*
* CHANGE ACTIVITY:
* $L0=DFSMS14,HDZ11D0,960628,SJPLMMR: DFM DataAgent initial code
*****/
#pragma csect (static, "DFMXAGN")
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <ctype.h>
#include <string.h>

#pragma csect (code, "DFMXAGNT")

int main(argc, argv)
    int argc;          /* count of input parameters */
    char **argv;        /* input parameters */
{
    int i;              /* loop counter */
    char dsname??(55??); /* data set name area */
    int debug = 1;      /* debug flag */

    /*****
    /* Define additional parameter for DataAgent special processing */
    /*****
    _Packed struct extra_parms {
        short int extra_parms_len; /* length of extra parms */
        /*****
        /* Basic section of the extra parameter structure */
        /*****
        unsigned short int reserved; /* reserved field */
        unsigned short int command_cp; /* command code point */
        unsigned short int object_cp; /* object code point */
        unsigned short int ofn_len; /* original filename length */
        char ofn??(54??); /* original filename */
        unsigned short int cfn_len; /* current/modified filename len */
        char cfn??(54??); /* current/modified filename */
        signed long int reason_code1; /* main reason code */
        signed long int reason_code2; /* secondary reason code */

```

Figure 32. DFM DataAgent Sample Routine Part 3 of 5

```

/*****
/* Any additions to the extra parameter structure in
/* future DFSMS releases would go here.
*****/
} ep_area; /* extra parameters for DataAgent
/* Define extra parameter instance
struct extra_parms *p_extra;

/*****
/* Begin DataAgent routine processing.
*****/
if (debug) {
/*****
/* Display input parameters for debugging purposes.
*****/
printf("DFMXAGNT: DataAgent routine entered.\n");
if (argc > 0) {
/*****
/* Display standard parameters and the program name.
*****/
printf("\n");
printf("Parameters passed were the following:\n");
for (i = 1; i < argc; i++)
printf(" %s\n",argv??(i??));
}
/*****
/* DataAgent has access to an extra parameter list in addition*/
/* to the standard format MVS parameter list. This parameter */
/* list is defined by the structure agent_parms.
*****/
/* Locate the extra parameters.
*****/
DFMXLPRM("DFMXAGNT",&p_extra);
if (p_extra == NULL) {
/*****
/* No parameters--something went wrong.
*****/
printf("DFMXAGNT: No DataAgent parameters!!\n");
return 8; /* exit with error
}

```

Figure 33. DFM DataAgent Sample Routine Part 4 of 5

```

if (p_extra->extra_parms_len >= sizeof(ep_area)) {
/*****
/* Print the basic section of the extra parameter list. */
/*****/
printf ("    \n");
printf ("Extra DataAgent parameters were:\n");
printf ("    DDM command code point: %X\n", p_extra->command_cp);
printf ("    DDM object code point: %X\n", p_extra->object_cp);
printf ("    DDM current filename length: %d\n",
        p_extra->cfn_len);
printf ("    DDM current filename: %s\n",
        strncpy(dsname, p_extra->cfn, p_extra->cfn_len));
printf ("    DDM original filename length: %d\n",
        p_extra->ofn_len);
printf ("    DDM original filename: %s\n",
        strncpy(dsname, p_extra->ofn, p_extra->ofn_len));
/*****/
/* Force recall of the exit (for DELDCL, etc.) */
/*****/
p_extra->reason_code1 = -1;
}
/* End, basic section exists */
/* End, parameters exist */
}
else {
/*****/
/* No parameters--something went wrong. */
/*****/
printf ("DFMXAGNT: No parameters?\n");
p_extra->reason_code1 = 2; /* set reason code */
return 8; /* exit with error */
}
/* End of missing parameters */
/* End of input parameter display */
}
/*****/
/* Any further DataAgent routine processing would go here. */
/*****/
return 0;
}

```

Figure 34. DFM DataAgent Sample Routine Part 5 of 5

DFMXSORT

System SAMPLIB member DFMXSORT, for example, SYS1.SAMPLIB(DFMXSORT) shown in [Figure 35](#) on [page 62](#), shows how a DFM DataAgent can be written in assembler language to invoke SORT.

```

      TITLE 'DFMXSORT - DFM DataAgent Sort Sample'
*/*****
*/**PROPRIETARY V3 STATEMENT**
*/**LICENSED MATERIALS - PROPERTY OF IBM**
*/**5695-DF1**
*/** (C) COPYRIGHT 1997  IBM CORP.**
*/**END PROPRIETARY V3 STATEMENT**
*/*****
*/**
*/** $MOD(DFMXSORT) COMP(5695-DF120)**
*/**
*/**  MODULE NAME: DFMXSORT**
*/**
*/**  DESCRIPTION: DFM DataAgent Sample Routine (SORT)**
*/**
*/**  STATUS: Version 1 Release 4.0 (DFSMS)**
*/**
*/**  COPYRIGHT: See copyright statement on previous page**
*/**
*/**  FUNCTION: This module illustrates how a DFM DataAgent can**
*/**    be written in assembler language to invoke SORT.  It is**
*/**    implemented with a basic function that assumes the input**
*/**    filename matches the SORTIN DD statement.  It changes the**
*/**    filename from SORTIN to SORTOUT so that any retrieval through**
*/**    DFM will retrieve the sorted data rather than the original.**
*/**
*/**    Code commentary is provided to show how the**
*/**    function could be extended to use the input filename for**
*/**    dynamic allocation and to derive an output filename.  The**
*/**    commentary also discusses how this DataAgent could request**
*/**    that it be called again at file close time to access and**
*/**    possibly write the output file's data to the original file.**
*/**
*/**  PROCESSING:**
*/**
*/**    LOGIC:**
*/**      Refer to block comments in the code.**
*/**
*/**    ERROR PROCESSING:**
*/**      Issue a WTO and then return with register 15 set to a**
*/**      non-zero value and with the reason code set to a unique**
*/**      value.  SORT error messages will be in the JOBLLOG.**
*/**
*/**

```

Figure 35. DFM DataAgent Sort Sample Part 1 of 7


```

**/ NOTES:
**/
**/ PATCH SPACE: None
**/ XAX CONSIDERATIONS: AMODE(31) RMODE(ANY) ENV(PRI)
**/ DEPENDENCIES: None
**/ RESTRICTIONS: None
**/ REGISTER CONVENTIONS: Standard
**/ SERIALIZATION: No serialization techniques are used by this
**/ module.
**/
**/ MODULE TYPE: Procedure
**/ PROCESSOR: z/OS Assembler
**/ ATTRIBUTES:
**/ TYPE: Not reentrant
**/ PRIMARY ASID: Caller's ASID
**/ SECONDARY ASID: Same as primary
**/ HOME ASID: Same as primary
**/ MODE: Task
**/ KEY: 8
**/ STATE: Problem program, non-APF-authorized
**/ LOCATION: Link library
**/
**/ ENTRY POINT: DFMXSORT
**/
**/ PURPOSE: Show that a DataAgent routine can invoke SORT.
**/ LINKAGE: Called by Distributed FileManager.
**/ INPUT: Refer to the DFM Guide and Reference for a general
**/ description of the parameter list format.
**/
**/ MESSAGES: Refer to WTO statements.
**/
**/ EXIT NORMAL:
**/
**/ RETURN CODE: Register 15 = 0
**/ REASON CODE: Not applicable
**/ MESSAGE ID: None:
**/
**/ EXIT ERRORS:
**/
**/ RETURN CODE: Register 15 = 8
**/ REASON CODE: Unique values set in the extended parameter
**/ list
**/ MESSAGE ID: See WTO statements.
**/
**/ EXTERNAL REFERENCES: None
**/
**/ CHANGE ACTIVITY:
**/ $LO=DFSMS14,HDZ11D0,960628,SJPLMMR: DFM DataAgent initial code
**/ *****
DFMXSORT AMODE 31
DFMXSORT RMODE ANY
DFMXSORT CSECT

```

Figure 36. DFM DataAgent Sort Sample Part 2 of 7

```

*****
*   This program is used as a DataAgent routine to get control   *
*   when a workstation's SdU application opens a remote MVS file *
*   with a filename suffix specifying "agent(dfmxsort)". The file is *
*   assumed to be in the format of the sample file created earlier. *
*   The input file is sorted and the filename is then changed to the *
*   filename of the sort output file. This results in the SdU      *
*   application's retrieving a sorted subset of the records as if   *
*   they came from the original file.                               *
*                                                                    *
*   An alternative invocation method is provided by SdU through    *
*   the DFMACALL sample application. Refer to it for details.      *
*                                                                    *
*   The reason code can be set to -1 to force entry at the file's   *
*   delete declaration time when adding extended function.         *
*****
R0      EQU      0
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
RTN     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
        SAVE      (14,12)      SAVE REGISTERS
        BALR      12,0          BRANCH AND LINK REG.
        USING     *,12          USE REG 12
        ST        13,SAVEAREA+4 SAVE BACKWARD POINTER
        LA        14,SAVEAREA   SET FORWARD PT.ER IN CALLER SAVE AREA
        ST        14,8(13)
        LR        13,14         SET OUR SAVE AREA
*-----*
*   Determine whether we are defining or deleting the file       *
*   declaration (i.e. previous to OPEN or after CLOSE).         *
*-----*
        LR        R3,R1         Save original parameter pointers
        SR        R15,R15       Clear error code
        USING     INPARMS,R3     Address of MVS parameter list
        L         R4,EXTPARMP    Point to DFM DataAgent parameters
        USING     EXTPARMS,R4    Base of DFM DataAgent parameters
        CLC       EXTBJCP,FILNAM Is file being processed?
        BNE       EXIT          No, exit

```

Figure 37. DFM DataAgent Sort Sample Part 3 of 7

```

*-----*
*      Called for a file-see whether declaration or delete      *
*      declaration.                                             *
*-----*
*      CLC      EXTCMDCP,DCLFIL  Is file being declared?
*      BE      DODCLFIL      Yes, process DCLFIL.
*      CLC      EXTCMDCP,DELDCL  Is file declaration being deleted?
*      BE      DODELDCL      Yes, process DELDCL.
*-----*
*      Unknown command type
*-----*
*      WTO      'DFMXSORT: Unknown command code.'
*      LA      R15,12
*      B      EXIT
*
*-----*
*      DCLFIL Processing
*-----*
DODCLFIL EQU *
*-----*
* Enhanced function:
* Set sort input filename from input filename.
* Set sort output filename to input filename.
* If output filename is greater than 40 characters then do.
*   Locate last component of filename.
*   If last component of filename is < 4 then
*     Locate last 2 components of filename.
*   If last component(s) are equal to ".SRT" then
*     Replace last component(s) with ".SR2".
*   Else
*     Replace last component(s) with ".SRT".
* end.
* Else
*   Append ".SRT" to output filename.
*   Allocate sort input as DISP=SHR.
*   Allocate sort output as DISP=(NEW,CATLG).
*-----*
*      Invoke DFSORT with 31-bit parameter list
*-----*
*      LR      R1,R3          Fetch address of std parm list
*      MVC     EXTPARMP,=F'-1' End of list
*      LINK    EP=SORT        Invoke DFSORT
*      LTR     R15,R15        Check for SORT failure
*      BNZ     SORTERR        Branch if error
*
*-----*
* Enhanced function:
* Set modified filename generated for output file allocation.
* Set reason code to -1 to force recall for DELDCL.
*-----*
*

```

Figure 38. DFM DataAgent Sort Sample Part 4 of 7

```

*-----*
*   Return modified filename and filename length to DFM.   *
*-----*
      LH   R2,EXTOFNLN      Get original (input) name length
      CH   R2,MAXIFNLN      Ensure maximum is not exceeded
      BH   OFNISOK          Branch-file name is left alone
      CH   R2,MINIFNLN      Ensure minimum is met
      BL   OFNISOK          Branch-file name is left alone
      LA   R1,EXTOFN        Point to beginning of orig fn
      AR   R1,R2            Point to end
      LH   R15,MINIFNLN     Get length of trigger in name
      SR   R1,R15           Backup to where trigger appears
      BCTR R15,0            Decrement for execute
      EX   R15,COMPNAME     Compare last part of name
      BNE  OFNISOK          Branch-no trigger at name end
*   Original file name meets qualifications-modify file name.
      LA   R3,EXTMFN        Set pointer to modified file name.
      LR   R1,R2            Set length of filename.
      SH   R1,MINIFNLN      Backup to trigger (=root end)
      BCTR R1,0            Decrement for execute
      EX   R1,MOVENAME      Move input name beginning to output
      LA   R1,1(R1)         Restore length
      AR   R3,R1            Point to end of root
      LH   R15,MODIFLEN     Get length of modifier
      BCTR R15,0            Decrement for execute
      EX   R15,CHANGENM     Change name to root + modifier
      SH   R2,MINIFNLN      Decrement trigger length
      AH   R2,MODIFLEN      Add modifier length
      STH  R2,EXTMFNLN      Set output name length
OFNISOK EQU   *            Here if changed/no change to do
      SR   R15,R15          Exit with no error
      B    EXIT            Exit DataAgent DCLFIL routine
COMPNAME CLC  0(0,R1),TRIGGER Compare name to trigger string
MOVENAME MVC  EXTMFN(0),EXTOFN Move original name to modified name
CHANGENM MVC  0(0,R3),MODIFIER Move modifier string to end of name
*
*-----*
*   SORT error occurred-refer to JOBLLOG for details.   *
*-----*
SORTERR EQU   *
      LR   R2,R15          Save SORT code
      ST   R0,EXTRSNC2     Save SORT reason code
      WTO  'DFMXSORT: SORT failure.'
      LR   R15,R2          Exit with error
      B    EXIT            Exit DataAgent routine
*

```

Figure 39. DFM DataAgent Sort Sample Part 5 of 7

```

*-----*
*      DELDCL Processing                                     *
*-----*
CODELDCL EQU      *
          WTO      'DFMXSORT: Can delete IBMUSER.DFMXSORT.SORTOUT now.'
*-----*
* Enhanced function:                                     *
*   Copy the changes to the permanent file.               *
*-----*
          SR      R15,R15      Exit with no error
          B       EXIT        Exit with return code from Sort
*-----*
*      Exit                                               *
*-----*
EXIT      EQU      *
          L       13,4(,13)    GET RETURN ADDRESS
          RETURN  (14,12),RC=(15) RESTORE REGS,FLAG SAVEAREA,SET RC
SAVEAREA DC      18F'00'
          LTORG
*
*-----*
*      Define input parameters.                           *
*-----*
*   In this case, the standard format MVS parameter list should be
*   a halfword length field followed by a DFSORT extended
*   parameter list. For example, Sort by ascending characters in
*   columns 17-22 is:
*       SORT FIELDS=(17,6,CH,A)
*-----*
*
*-----*
*      Constants for use with the DFM DataAgent extended parameters. *
*-----*
DCLFIL    DC      X'102C'      Declare file command code point
DELDCL    DC      X'102D'      Delete declare file command code point
DRCNAM    DC      X'1165'      Directory is being declared
FILNAM    DC      X'110E'      File is being declared
*-----*
*      Local Constants                                     *
*-----*
MAXIFNLN  DC      H'53'        Maximum allowing for trigger->modifier
MINIFNLN  DC      H'6'         Minimum orig name len (= len of trigger)
TRIGGER    DC      CL6'SORTIN'  Trigger in input filename
MODIFLEN   DC      H'7'         Length of modifier
MODIFIER   DC      CL7'SORTOUT' Modifier for output filename

```

Figure 40. DFM DataAgent Sort Sample Part 6 of 7

```

*-----*
*   Parameter list pointers.                               *
*-----*
INPARMS DSECT
STDPARMP DS   A(0)          Ptr to standard format MVS parameter list
EXTPARMP DS   A(0)          Ptr to extended DFM parameter list
*
*-----*
*   Standard MVS parameter list for SORT usage.           *
*-----*
STDPARMS DSECT
STDPARML DS   H             Length of parameters
STDPARMC DS   CL256         Standard parameter string
*
*-----*
*   Extended parameter list unique to DFM.                 *
*-----*
EXTPARMS DSECT
EXTPARML DS   H             Length of parameters
           DS   H             Reserved
EXTCMDCP DS   H             Command code point
EXTOBJCP DS   H             Object code point
EXTOFNLN DS   H             Original filename length
EXTOFN   DS   CL54          Original filename
EXTMFNLN DS   H             Modified filename length
EXTMFN   DS   CL54          Modified filename
EXTRSN1  DS   F             Reason code 1
EXTRSN2  DS   F             Reason code 2
*
          END   DFMXSORT

```

Figure 41. DFM DataAgent Sort Sample Part 7 of 7

DFMXSRTI

System SAMPLIB member DFMXSRTI, for example, SYS1.SAMPLIB(DFMXSRTI) shown in [Figure 42](#) on [page 69](#), shows how to install the DFMXSORT DataAgent routine.

```

//DFMXSRTI JOB , 'DFMXSORT SETUP',MSGLEVEL=(1,1),MSGCLASS=A,
//          USER=IBMUSER,PASSWORD=IBM,REGION=1M
//*****
//*PROPRIETARY V3 STATEMENT
//*LICENSED MATERIALS - PROPERTY OF IBM
//*5695-DF1
//*(C) COPYRIGHT 1997 IBM CORP.
//*END PROPRIETARY V3 STATEMENT
//*****
//* Sample installation for DFMXSORT DataAgent routine. Modify the
//* job statement, etc. as appropriate for your installation.
//*****
//* This sample uses &PREFIX to generate input and output filenames.
//* For example, &PREFIX.SORTIN will be used for input and
//* &PREFIX.SORTOUT will be used for output.
//*
//* A sample input file, IBMUSER.DFMXSORT.SORTIN, is also provided.
//*
//* It could be cloned with other filenames and sort parameters
//* or could be generalized to use dynamic allocation for the
//* input and output files. Pseudocode is included to illustrate
//* the types of changes that would be required. If changes are
//* made, then you must remove DD statements for SORTIN and
//* SORTOUT from the cataloged procedure.
//*
//* Once this installation job has run, logon to a workstation with
//* the prerequisite level of SdU and with an APPC connection to the
//* mainframe and issue the following command:
//* dfmacall agent x:ibmuser.dfmxsrt.sortin
//*          'dfmxsort,prefix=ibmuser.dfmxsrt'
//*          parm 'sort fields=(17,6,ch,a)'
//*
//* Refer to DFMACALL for details.
//*
//*****

```

Figure 42. Sample Installation for the DFMXSORT DataAgent Routine Part 1 of 4

```

//*****
//* Clean up test data sets. Remove this step from production version.
//*****
//CLEANUP      EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
      DELETE ( -
              IBMUSER.DFMXSORT.* -
              ) NONVSAM PURGE SCRATCH
      IF LASTCC = 8 THEN DO
        SET LASTCC = 0
        SET MAXCC = 0
      END
/*

//*****
//*      Generate SORTIN and allocate SORTOUT      */
//*****
//STEP1      EXEC PGM=IEBGENER
//SYSUT2     DD
DSN=IBMUSER.DFMXSORT.SORTIN,DISP=(NEW,CATLG),UNIT=SYSDA,
//          DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//          SPACE=(TRK,(1,1))
//SORTOUT    DD DSN=IBMUSER.DFMXSORT.SORTOUT,DISP=(NEW,CATLG),
//
UNIT=SYSDA,DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//          SPACE=(TRK,(1,1))
//* Create records with a 6 character sort field in column 17
//* and with a flag in column 40 that is used to omit (=1) or
//* include the record. The program itself contains the parameters
//* for the sort (ascending on columns 17-22) and the control file
//* contains the definition of what records to include/omit.
//SYSUT1     DD DATA
RECORD NUMBER = 000030, OMIT FLAG IS = 1 .....
RECORD NUMBER = 000888, OMIT FLAG IS = 0 .....
RECORD NUMBER = 000887, OMIT FLAG IS = 1 .....
RECORD NUMBER = 099999, OMIT FLAG IS = 0 .....
RECORD NUMBER = 100000, OMIT FLAG IS = 1 .....
RECORD NUMBER = 100001, OMIT FLAG IS = 0 .....
RECORD NUMBER = 111111, OMIT FLAG IS = 1 .....
RECORD NUMBER = 111110, OMIT FLAG IS = 1 .....
RECORD NUMBER = 211111, OMIT FLAG IS = 0 .....
RECORD NUMBER = 999999, OMIT FLAG IS = 1 .....
RECORD NUMBER = 000000, OMIT FLAG IS = 0 .....
/*
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD DUMMY

```

Figure 43. Sample Installation for the DFMXSORT DataAgent Routine Part 2 of 4


```

//*****
//* Assemble the DataAgent Sample DFMXSORT
//*****
//ASM01 EXEC PGM=IEV90,PARM='OBJECT,NODECK'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSLIN DD DSN=&&DFMXSORT,DISP=(,PASS),
// UNIT=SYSDA,SPACE=(TRK,(2,2,2))
//SYSIN DD DSN=SYS1.SAMPLIB(DFMXSORT),DISP=SHR
//*****
//* Link Edit DataAgent Routine DFMXSORT *
//*****
//LINK1 EXEC
PGM=IEWL,PARM='XREF,LET,LIST,AMODE=31,RMODE=ANY'
//SYSPRINT DD SYSOUT=*
//OBJ DD DSN=&&DFMXSORT,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=SYS1.LINKLIB(DFMXSORT),DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(80,10))
//SYSLIN DD *
INCLUDE OBJ
ENTRY DFMXSORT
NAME DFMXSORT(R)

/*
//*****
//* Build Agent JCL in SYS1.PROCLIB
//*****
//STEP1 EXEC PGM=IEBGENER
//SYSUT2 DD DISP=SHR,DSN=SYS1.PROCLIB(DFMXSORT)
//SYSUT1 DD DATA
//DFMXSORT JOB ,MSGCLASS=A
//*****
//* Set appropriate msgclass above for debug vs production
//*****
//DFMXSORT PROC DFMINIT=,PREFIX=IBMUSER.DFMXSORT
//DFMAGENT EXEC PGM=DFMINIT
//*****
//* Run DFMXSORT DataAgent Sample
//*
//* Sort input comes from IBMUSER.DFMXSORT.SORTIN by default.
//* Sorted data goes into IBMUSER.DFMXSORT.SORTOUT by default.
//* You should modify the names, space allocation, etc. as appropriate
//* for your installation.

```

Figure 44. Sample Installation for the DFMXSORT DataAgent Routine Part 3 of 4

```

//*****
//* Add STEPLIB statements as appropriate for your installation.
//*STEPLIB DD DSN=...,DISP=SHR
//* DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=&PREFIX..SORTIN,DISP=SHR
//SORTOUT DD DSN=&PREFIX..SORTOUT,DISP=SHR
//SORTCNTL DD DSN=SYS1.PROCLIB(DFMXSORI),DISP=SHR
//DFMXSORT PEND
//GO EXEC DFMXSORT
/*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*****
//* Build omit control statement in SYS1.PROCLIB
//*****
//STEP2 EXEC PGM=IEBGENER
//SYSUT2 DD DISP=SHR,DSN=SYS1.PROCLIB(DFMXSORI)
//SYSUT1 DD DATA
* The following omit statement will ensure that only the
* records without a 1 in column 40 appear in the sortout data set
OMIT COND=(40,4,CH,EQ,C'1')
/*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY

```

Figure 45. Sample Installation for the DFMXSORT DataAgent Routine Part 4 of 4

DFMQTSO

System SAMPLIB member DFMQTSO, for example, SYS1.SAMPLIB(DFMQTSO) shown in [Figure 46 on page 72](#), shows how a DFM DataAgent can be written in assembler language to invoke TSO. The DFMQTSO routine links to the IKJTSOEV function.

```
      TITLE 'DFMQTSO - DFM DataAgent TSO Sample'
      */*****
      */**PROPRIETARY V3 STATEMENT
      */**LICENSED MATERIALS - PROPERTY OF IBM
      */**5695-DF1
      */** (C) COPYRIGHT 1997  IBM CORP.
      */**END PROPRIETARY V3 STATEMENT
      */*****
      */*****
      */** $MOD(DFMQTSO) COMP(5695-DF120)
      */**
      */**  MODULE NAME: DFMQTSO (Quick TSO-Input in PARM)
      */**
      */**  DESCRIPTION: DFM DataAgent Sample Routine (TSO)
      */**
      */**  STATUS: Version 1 Release 4.0 (DFSMS)
      */**
      */**  COPYRIGHT: See copyright statement on previous page
      */**
      */**  FUNCTION: This module illustrates how a DFM DataAgent can
      */**    be written in assembler language to invoke TSO.
      */**
      */**    SdU provides a sample application, DFMACALL, that can be
      */**    to invoke this sample.  Refer to it for details.
      */**
      */**    Refer to 'TSO Extensions for MVS: Programming Services' for
      */**    information about using the TSO environment service used in
      */**    this example.
      */**
      */**  PROCESSING:
      */**
      */**    LOGIC:
      */**      Refer to block comments in the code.
      */**
      */**    ERROR PROCESSING:
      */**      Issue a WTO and then return with register 15 set to a
      */**      non-zero value and with the reason code set to a unique
      */**      value.
      */**
```

Figure 46. DFM DataAgent Sample Routine (TSO) Part 1 of 7

```

**/ NOTES:
**/
**/ PATCH SPACE: None
**/ XAX CONSIDERATIONS: AMODE(31) RMODE(ANY) ENV(PRI)
**/ DEPENDENCIES: None
**/ RESTRICTIONS: None
**/ REGISTER CONVENTIONS: Standard
**/ SERIALIZATION: No serialization techniques are used by this
**/ module.
**/
**/ MODULE TYPE: Procedure
**/ PROCESSOR: z/OS Assembler
**/ ATTRIBUTES:
**/ TYPE: Not reentrant
**/ PRIMARY ASID: Caller's ASID
**/ SECONDARY ASID: Same as primary
**/ HOME ASID: Same as primary
**/ MODE: Task
**/ KEY: 8 (Current task TCBPKF=jobstep task TCBPKF)
**/ STATE: Problem program, non-APF-authorized
**/ LOCATION: Link library
**/
**/ ENTRY POINT: DFMQTSO
**/
**/ PURPOSE: Show that a DataAgent routine can invoke TS0.
**/ LINKAGE: Called by Distributed FileManager.
**/ INPUT: Refer to the DFM Guide and Reference for a general
**/ description of the parameter list format. Refer to
**/ DFMACALL documentation for command line invocation from
**/ SdU.
**/
**/ MESSAGES: Refer to WTO statements.
**/
**/ EXIT NORMAL:
**/
**/ RETURN CODE: Register 15 = 0
**/ REASON CODE: Not applicable
**/ MESSAGE ID: None:
**/
**/ EXIT ERRORS:
**/
**/ RETURN CODE: Register 15 = non-zero
**/ REASON CODE: Unique values set in the extended parameter
**/ list
**/ MESSAGE ID: See WTO statements.
**/
**/ EXTERNAL REFERENCES: None
**/
**/ CHANGE ACTIVITY:
**/ $L0=DFSMS14,HDZ11D0,960628,SJPLMMR: DFM DataAgent initial code
**/ *****

```

Figure 47. DFM DataAgent Sample Routine (TSO) Part 2 of 7

```

DFMQTSO CSECT
DFMQTSO AMODE 31
DFMQTSO RMODE ANY
        STM R14,R12,12(R13)
        BALR R12,0
        USING *,R12
        ST R13,SAVEAREA+4
        LA R11,SAVEAREA
        ST R11,8(R13)
        LA R13,SAVEAREA
        LR R3,R1          Save original parameter pointers
        SR R15,R15        Clear error code
        USING INPARMS,R3  Address of MVS parameter list
        L R4,STDPARM      Point to DFM DataAgent parameters
        USING STDPARMS,R4 Base of DFM DataAgent parameters
        L R5,EXTPARMP     Point to DFM DataAgent parameters
        USING EXTPARMS,R5 Base of DFM DataAgent parameters
*-----*
*      Called for a file or directory.  See whether declaration      *
*      is being created or deleted.                                  *
*-----*
        CLC EXTCMDCP,DCLFIL Is file being declared?
        BE DODCLFIL        Yes, process DCLFIL.
        CLC EXTCMDCP,DELDCL Is file declaration being deleted?
        BE DODELDCL        Yes, process DELDCL.
*-----*
*      Unknown command type                                         *
*-----*
        WTO 'DFMQTSO: Unknown command code.'
        LA R15,16          Exit without trying to set reason code
        B EXIT
*
*-----*
*      DCLFIL Processing (DCLFIL, or Declare File, is a DDM,      *
*      Distributed Data Management, command issued when a remote   *
*      file or directory is about to be opened)                    *
*-----*
DODCLFIL EQU *
        WTO 'DFMQTSO: Declaring a file.'
*****
* CALTSOEV - CALL THE TSO/E ENVIRONMENT SERVICE TO ESTABLISH A TSO/E
*      ENVIRONMENT IN THIS PROGRAM'S ADDRESS SPACE.
*      PARM1 IS RESERVED
*      PARM2 IS A FULLWORD THAT WILL CONTAIN THE RETURN CODE FROM IKJTISOEV
*      PARM3 IS A FULLWORD THAT WILL CONTAIN THE REASON CODE ON RETURN
*      FROM IKJTISOEV.
*      PARM4 IS A FULLWORD THAT WILL CONTAIN THE ABEND CODE, IF AN ABEND
*      OCCURS DURING TSO/E ENVIRONMENT SERVICE PROCESSING.
*      PARM5 IS A FULLWORD THAT WILL CONTAIN THE ADDRESS OF THE CPPL.
*****
CALTSOEV DS 0H
        XC PARM1,PARM1
        LINK EP=IKJTISOEV,ERRET=LE,PARAM=(PARM1,PARM2,PARM3,PARM4,PARM*
        5),VL=1

```

Figure 48. DFM DataAgent Sample Routine (TSO) Part 3 of 7

```

*****
* CHKEVRC - CHECK THE RETURN CODE FROM IKJTSOEV
*****
CHKEVRC  DS      0H
         L        R2,PARM2
         LTR      R2,R2
         BNZ      BADEVRC
*****
* TSO Environment established-process the input file.
*****
* CALLTSR - Call IKJEFTSR to process the input file or the parameter
*           list-dependent on whether a parameter list is present.
*           The output from the commands will go to the SYSTSPRT file.
*****
CALLTSR  DS      0H
         LH       R2,STDPARML           Get length of input parameters
         LTR      R2,R2                 If zero then use the input file
         BZ       USEINFIL
         ST       R2,BUFLen           Set buffer length
         BCTR     R2,0
         EX       R2,COPYCMD           Copy command to parameter area
         L        R15,CVTPTR
         L        R15,CVTTVT(,R15)
         L        R15,TSVTASF-TSVT(,R15)
         CALL     (15),(FLAGS,CMDBUFF,BUFLen,RETCODE,RSNCODE,ABNDCODE),VL
*****
* DOALL - At this point, process the return values from
*         IKJEFTSR and the invoked functions.
*****
DOALL    DS      0H
         LTR      R15,R15
         BZ       EXIT                 Exit if no error
         C        R15,=F'4'           Did CLIST or REXX exec fail?
         BNE      SAVERC              No, just ensure RC is saved
         MVC      RSNCODE,RETCODE     Use CLIST/REXX RC as reason code
SAVERC   ST       R15,RETCODE         Ensure a retcode is set
         WTO      'DFMQTSO: IKJEFTSR failed.'
         L        R15,RETCODE         Set return code
         MVC      EXTRSNC1,RSNCODE    Set reason code 1
         MVC      EXTRSNC2,ABNDCODE   Set reason code 2 to ABEND code
         B        EXIT
COPYCMD  MVC      CMDBUFF(0),STDPARMC
*

```

Figure 49. DFM DataAgent Sample Routine (TSO) Part 4 of 7

```

*-----*
*      SYSTSIN Processing                                     *
*-----*
USEINFIL EQU      *
      WTO      'DFMQTSO: SYSTSIN input not supported. Use DFMXTSO.'
      LA       R15,8
      B        EXIT

*
*-----*
*      DELDCL Processing (DELDCL, or Delete Declaration, is a DDM, *
*      Distributed Data Management, command issued after a remote *
*      file or directory has been closed)                         *
*-----*
DODELDCL EQU      *
      WTO      'DFMQTSO: Deleting file declaration.'
      SR       R15,R15      Exit with no error
      B        EXIT      Exit with return code from Sort
*****
* LE      - Branch here if LINK failed. The ABEND code will be in
*           register 1 but there will be no reason code. Set the
*           return code to 8 and use the ABEND code as the reason code.
*****
*
LE      DS      0H
      ST       R1,EXTRSN1      Save the ABEND code
      WTO      'DFMQTSO: LINK to IKJTSOEV failed.'
      LA       R15,8
      B        EXIT

*
*****
* BADEVRC - Branch here if IKJTSOEV returned a non-zero return code.
*           If the program branches here, it will exit with an error.
*           In the DFM diagnostics, the error data will be as follows:
*           RETURN CODE - THE RETURN CODE FROM IKJTSOEV
*           REASON CODE 1 - THE REASON CODE FROM IKJTSOEV
*           REASON CODE 2 - THE ABEND CODE FROM IKJTSOEV
*****
BADEVRC DS      0H
      WTO      'DFMQTSO: IKJTSOEV error occurred.'
      L        R15,PARM2
      MVC      EXTRSN1,PARM3
      MVC      EXTRSN2,PARM4
      B        EXIT

```

Figure 50. DFM DataAgent Sample Routine (TSO) Part 5 of 7

```

*****
* EXIT - RETURN TO CALLING PROGRAM
*****
EXIT      DS      0H
          L        R13,4(,R13)
          RETURN  (14,12),RC=(15) RESTORE REGS,FLAG SAVEAREA,SET RC
*
* REGISTER EQUATES
R1        EQU     1
R2        EQU     2
R3        EQU     3
R4        EQU     4
R5        EQU     5
R11       EQU     11
R12       EQU     12
R13       EQU     13
R14       EQU     14
R15       EQU     15
* PARAMETERS USED TO INVOKE THE TSO/E ENVIRONMENT SERVICE
PARM1     DS      F              RESERVED FIELD
PARM2     DS      F              RETURN CODE FIELD
PARM3     DS      F              REASON CODE FIELD
PARM4     DS      F              FUNCTION ABEND CODE
PARM5     DS      F              CPPL ADDRESS
* PARAMETERS USED TO INVOKE THE TSO SERVICE FACILITY
FLAGS     DS      0F              FULLWORD OF FLAGS
RESFLAGS  DC      H'0001'        ESTABLISH UNAUTHORIZED ENVIRONMENT
ABFLAGS   DC      X'01'          PRODUCE A DUMP IF FUNCTION ABENDS
FNCFLAGS  DC      X'01'          INVOKE TSO/E CMD, REXX EXEC, CLIST
CMDBUFF   DS      256C           COMMAND BUFFER
BUFLLEN   DS      F              LENGTH OF COMMAND BUFFER
RETCODE   DS      F              FUNCTION RETURN CODE
RSNCODE   DS      F              FUNCTION REASON CODE
ABNDCODE  DS      F              FUNCTION ABEND CODE
CVTPTR    EQU     16             THESE 2 PARMS ARE USED TO GET
CVTTVT    EQU     X'9C'          ADDR OF THE TSO SERVICE FACILITY
* SAVEAREA AND OTHER PROGRAM STORAGE
SAVEAREA  DS      18F
* TSVT MAPPING MACRO (USED TO GET THE ADDRESS OF TSO SERVICE FACILITY)
          IKJTSVT
DFMQTSO   CSECT
          LTORG
*
*-----*
*   Define input parameters.                               *
*-----*
*   The standard format MVS parameter list is a halfword length *
*   field followed by a parameter list. In this case the      *
*   parameter list is a TSO command.                          *
*-----*
*

```

Figure 51. DFM DataAgent Sample Routine (TSO) Part 6 of 7

```

*-----*
*   Constants for use with the DFM DataAgent extended parameters. *
*-----*
DCLFIL  DC   X'102C'      Declare file command code point
DELDCL  DC   X'102D'      Delete declare file command code point
DRCNAM  DC   X'1165'      Directory is being declared
FILNAM  DC   X'110E'      File is being declared
*-----*
*   Parameter list pointers. *
*-----*
INPARMS DSECT
STDPARMP DS   A(0)        Ptr to standard format MVS parameter list
EXTPARMP DS   A(0)        Ptr to extended DFM parameter list
*
*-----*
*   Standard MVS parameter list for SORT usage. *
*-----*
STDPARMS DSECT
STDPARML DS   H           Length of parameters
STDPARMC DS   CL256       Standard parameter string
*
*-----*
*   Extended parameter list unique to DFM. *
*-----*
EXTPARMS DSECT
EXTPARML DS   H           Length of parameters
          DS   H           Reserved
EXTCMDCP DS   H           Command code point
EXTOBJCP DS   H           Object code point
EXTOFNLN DS   H           Original filename length
EXTOFN   DS   CL54        Original filename
EXTMFNLN DS   H           Modified filename length
EXTMFN   DS   CL54        Modified filename
EXTRSNC1 DS   F           Reason code 1
EXTRSNC2 DS   F           Reason code 2
*
          END      DFMQTSO

```

Figure 52. DFM DataAgent Sample Routine (TSO) Part 7 of 7

DFMXTSOI

System SAMPLIB member DFMXTSOI, for example, SYS1.SAMPLIB(DFMXTSOI) shown in [Figure 53](#) on [page 79](#), shows how to install the two DataAgent routines: DFMXTSO and DFMQTSO.


```

//DFMXTSOI JOB , 'DFMXTSO SETUP',MSGLEVEL=(1,1),MSGCLASS=A,
//          USER=IBUSER,PASSWORD=IBM,REGION=1M
//*****
//*PROPRIETARY V3 STATEMENT
//*LICENSED MATERIALS - PROPERTY OF IBM
//*5695-DF1
//*(C) COPYRIGHT 1997 IBM CORP.
//*END PROPRIETARY V3 STATEMENT
//*****
//* Install two TSO DataAgent routines, DFMXTSO and DFMQTSO by
//* installing two procedures DFMXTSO and DFMQTSO and by producing
//* a load module for DFMQTSO. No executable is produced for
//* procedure (or DataAgent) DFMXTSO because it uses the standard
//* TSO batch program, IKJEFT01.
//*
//* DFMXTSO is full function in that it is intended to be used with
//* IKJEFT01 which accepts input from SYSTSIN. DFMQTSO (or quick
//* TSO) only accepts input from the PARM() parameter.
//*
//* Once this installation job has run, DFMACALL TSO or DFMACALL QTSO
//* can be run from a workstation that has SdU installed and that has
//* an APPC connection to the mainframe. Refer to DFMACALL for
//* details. (Note that you might want to set up some typical
//* SYSTSIN files and preallocate several SYSTSPRT files for the
//* anticipated users.)
//*

```

Figure 53. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 1 of 7

```

//*****
//* General setup
//*****
//*****
//* Clean up old SYSTSIN and SYSTSPRT files.
//* (Remove or modify if copying this to production JCL.)
//*****
//CLEANUP EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE ( -
        IBMUSER.DFMXTSO.SYSTSIN -
        ) NONVSAM PURGE SCRATCH
DELETE ( -
        IBMUSER.DFMQTSO.SYSTSPRT -
        ) NONVSAM PURGE SCRATCH
DELETE ( -
        IBMUSER.DFMXTSO.SYSTSPRT -
        ) NONVSAM PURGE SCRATCH
IF LASTCC = 8 THEN
DO
SET LASTCC = 0
SET MAXCC = 0
END

```

Figure 54. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 2 of 7

```

/*
//*****
//*   Allocate output files                               */
//*   (Remove or modify if copying this to production JCL.) */
//*****
//ALLOCATE EXEC PGM=IEFBR14
//ALLOC1 DD DSN=IBMUSER.DFMQTSO.SYSTSPRT,DISP=(NEW,CATLG),UNIT=SYSDA,
//          DCB=(DSORG=PS,LRECL=121,BLKSIZE=0,RECFM=FBA),
//          SPACE=(TRK,(15,5))
//ALLOC2 DD DSN=IBMUSER.DFMXTSO.SYSTSPRT,DISP=(NEW,CATLG),UNIT=SYSDA,
//          DCB=(DSORG=PS,LRECL=121,BLKSIZE=0,RECFM=FBA),
//          SPACE=(TRK,(15,5))
//*****
//*   Generate sample input file for full function DFMXTSO */
//*   (Remove or modify if copying this to production JCL.) */
//*****
//GENINPUT EXEC PGM=IEBGENER
//SYSUT2 DD DSN=IBMUSER.DFMXTSO.SYSTSIN,DISP=(NEW,CATLG),UNIT=SYSDA,
//          DCB=(LRECL=80,BLKSIZE=0,RECFM=FB),
//          SPACE=(TRK,(1,1))
//SYSUT1 DD DATA
//          /* This is a sample input file for SYSTSIN.          */
//          LISTC
/*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*/

```

Figure 55. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 3 of 7

```

//*****
//* 1. Setup for the DFMXTSO DataAgent procedure. Modify the job
//* statement, etc. as appropriate for your installation.
//*****
//* This setup job installs a sample procedure, DFMXTSO, to allow
//* workstations to invoke TSO.
//*
//* The procedure is intended to be used with PGM(IKJEFT01) and will
//* accept input from both the PARM field (processed first) and
//* from userID.DFMXTSO.SYSTSIN.
//*
//* It uses userID.DFMXTSO.SYSTSPRT to contain the TSO output.
//*
//* Note that SYSTSPRT allocation is DISP=SHR.
//* This causes SYSTSPRT to be reset each invocation
//* of TSO so the output produced by a file declaration
//* will be overlain by a subsequent delete file declaration.
//* Therefore, end users or applications will have to use the
//* SYSTSPRT file contents before closing the file replaces it.
//* The SYSTSPRT allocation could also be changed to DISP=MOD to
//* cause appending of output.
//*
//* You must create the SYSTSPRT or SYSTSIN files that
//* will be needed before invoking the exit. If parameters are
//* to be passed through the PARM field only, SYSTSIN could
//* be changed to DD DUMMY or the input file could be cleared.
//*
//* The exit must be invoked with at least the following
//* parameters: ...,AGENT(DFMXTSO,U=userID),PARM(...)
//*
//* (DFM for z/OS will automatically provide the U=userID parameter)
//*
//*****

```

Figure 56. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 4 of 7

```

//*****
//* Build Agent JCL in SYS1.PROCLIB
//*****
//GENPROC1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT2 DD DISP=SHR,DSN=SYS1.PROCLIB(DFMXTS0)
//SYSUT1 DD DATA
//DFMXTS0 JOB ,MSGCLASS=A
//*****
//* Set appropriate msgclass above for debug vs production
//* Note that DFM will provide the DFMINIT parameter and the
//* U parameter (high-level name qualifier or userID).
//*****
//DFMXTS0 PROC DFMINIT=,U=
//DFMAGENT EXEC PGM=&DFMINIT.,
//          PERFORM=2,
//          REGION=5000K,
//          DYNAMNBR=20
//*****
//* Run DFMXTS0 DataAgent Sample
//*****
//* Add STEPLIB statements as appropriate for your installation.
//* Note that if IKJEFT01 was installed into LPALIB, a STEPLIB will
//* be required because DFM's DataAgent processing uses the BLDL
//* function.
//*STEPLIB DD DSN=...,DISP=SHR
//*          DD DSN=SYS1.LINKLIB,DISP=SHR
//*          DD DSN=SYS1.LPALIB,DISP=SHR  <-- See note above
//*
//* TS0/E input comes from the SYSTSIN file (as well as from PARM).
//*
//* TS0/E output goes to the SYSTSPRT file.
//*
//* Sample using a generic CLIST...
//*SYSPROC DD DSN=&U..CLIST.CLIST,DISP=SHR
//SYSPROC DD DSN=SYS1.CLIST,DISP=SHR
//SYSTSIN DD DSN=&U..DFMXTS0.SYSTSIN,DISP=SHR
//SYSTSPRT DD DSN=&U..DFMXTS0.SYSTSPRT,DISP=SHR
//SYSOUT DD SYSOUT=*
//DFMXTS0 PEND
//GO EXEC DFMXTS0
/*

```

Figure 57. Sample Installation for the DFMXTS0 and DFMQTS0 DataAgent Routines Part 5 of 7

```

//*****
//* 2. Setup for the DFMQTSO DataAgent procedure.  Modify the job
//* control statements, etc. as appropriate for your installation.
//*****
//* This setup job installs a sample procedure, DFMQTSO, to allow
//* workstations to invoke program DFMQTSO which will, in turn, invoke
//* IKJTSEV to establish the TSO environment and then call TSO to
//* process the input from the PARM field.  It ignores file
//* userID.DFMXTSO.SYSTSIN.
//*
//* Like DFMXTSO, it uses userID.DFMXTSO.SYSTSPRT to contain
//* the TSO output.
//*
//* The exit must be invoked with at least the following
//* parameters:  ...,AGENT(DFMQTSO,U=userID),PARM(...)
//*
//* (DFM for z/OS will automatically provide the U=userID parameter)
//*
//*****
//*****
//* Assemble the Quick TSO Sample DataAgent Routine, DFMQTSO
//*****
//ASM01 EXEC PGM=IEV90,PARM='OBJECT,NODECK'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSLIN DD DSN=&&DFMQTSO,DISP=(,PASS),
// UNIT=SYSDA,SPACE=(TRK,(2,2,2))
//SYSIN DD DSN=SYS1.SAMPLIB(DFMQTSO),DISP=SHR
//*****
//* Link Edit DataAgent Routine DFMQTSO *
//*****
//LINK1 EXEC PGM=IEWL,PARM='XREF,LET,LIST,AMODE=31,RMODE=ANY'
//SYSPRINT DD SYSOUT=*
//OBJ DD DSN=&&DFMQTSO,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=SYS1.LINKLIB(DFMQTSO),DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(80,10))
//SYSLIN DD *
// INCLUDE OBJ
// ENTRY DFMQTSO
// NAME DFMQTSO(R)
/*

```

Figure 58. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 6 of 7

```

//*****
//* Build DFMQTSO Agent JCL in SYS1.PROCLIB
//*****
//GENPROC2 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT2 DD DISP=SHR,DSN=SYS1.PROCLIB(DFMQTSO)
//SYSUT1 DD DATA
//DFMQTSO JOB ,MSGCLASS=A
//*****
//* Set appropriate msgclass above for debug vs production
//* Note that DFM will provide the DFMINIT parameter and the
//* U parameter (high-level name qualifier or userID).
//*****
//DFMQTSO PROC DFMINIT=,U=
//DFMAGENT EXEC PGM=&DFMINIT.,
//          PERFORM=2,
//          REGION=5000K,
//          DYNAMNBR=20
//*****
//* Run DFMQTSO DataAgent Sample
//*****
//* Add STEPLIB statements as appropriate for your installation.
//*STEPLIB DD DSN=...,DISP=SHR
//*          DD DSN=SYS1.LINKLIB,DISP=SHR
//*
//* Sample shown using a global CLIST library.
//*SYSPROC DD DSN=&U..CLIST.CLIST,DISP=SHR
//SYSPROC DD DSN=SYS1.CLIST,DISP=SHR
//*
//* TS0/E output goes to the SYSTSPRT file.
//*
//SYSTSPRT DD DSN=&U..DFMQTSO.SYSTSPRT,DISP=SHR
//*
//* Note that the program DFMQTSO does not use SYSTSIN for input
//* so a dummy SYSTSIN DD statement is provided.
//*
//SYSTSIN DD DUMMY
//SYSOUT DD SYSOUT=*
//DFMQTSO PEND
//GO EXEC DFMQTSO
//*

```

Figure 59. Sample Installation for the DFMXTSO and DFMQTSO DataAgent Routines Part 7 of 7

DFM

System PROCLIB member DFM, for example, SYS1.PROCLIB(DFM), contains the sample shown in [Figure 60 on page 84](#) of the startup procedures for Distributed FileManager. See [“Activating Distributed FileManager in System PROCLIB” on page 41](#).

Note: LE is required to use CDRA, if LE is installed and is not in the link list, SYS1.PROCLIB(DFM) and SYS1.SAMPLIB(GDETPDEF) should be modified so their STEPLIB DD statements refer to the proper LE runtime library. Refer to DFMREADM in SYS1.SAMPLIB for details.

```

//DFM      PROC  PARMS='NORMAL'
//*****
//*
//*  DFSMS DFM START UP PROCEDURE
//*
//*****
//DFM      EXEC  PGM=GDEISBOT,
//          PARM='&PARMS&sssq;',
//          REGION=0K,
//          TIME=1440
//IEFPARM  DD   DSN=SYS1.PARMLIB,DISP=SHR
//* CHANGE THE STEPLIB STATEMENT AS REQUIRED IF YOUR INSTALLATION
//* DOES NOT HAVE THE LE RUNTIME DATA SET IN ITS LINK LIST.
//*STEPLIB DD   DSN=SYS1.SCEERUN,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//*
//*  THE TWO FILES ASSOCIATED WITH THE DD STATEMENTS CDRATRC AND
//*  SYSOUT CAN BE USED TO DIAGNOSE DFM STARTUP PROBLEMS RELATED
//*  TO CDRA. (CDRA IS INVOKED DURING STARTUP FOR CERTAIN CCSID
//*  VALUES IN THE SYS1.PARMLIB MEMBER DFM00.)
//*
//*  YOU MUST ALLOCATE THE TWO FILES AS RECFM=FBA, LRECL=133,
//*  AND DSORG=PS BEFORE STARTING DFM WITH THE DD STATEMENTS
//*  ACTIVE.
//*
//*  NOTE THAT SYSOUT IS REQUIRED AND CDRATRC IS OPTIONAL
//*  WHEN USING CDRA AND THE DEFAULT INSTALLATION IS SET UP TO
//*  USE CDRA IF YOUR HOST CODE PAGE IS OTHER THAN 500.
//*
//*  CDRATRC DD   DSN=SYS1.CDRATRC,DISP=SHR    CDRA API TRACING
//*  SYSOUT  DD   DSN=SYS1.CDRAOUT,DISP=SHR    C RUNTIME MESSAGES
//SYSOUT   DD   DUMMY <- DEFAULT = CDRA WITH RUNTIME MESSAGES DISCARDED

```

Figure 60. DFM Startup Procedure

Chapter 6. PPT Entries for Distributed FileManager

Figure 61 on page 85 shows the PPT entries for Distributed FileManager (see [“Verifying PPT Entries for Distributed FileManager”](#) on page 41).

```
PPT PGMNAME(GDEISASB) /* DFM CONVERSATION ADDRESS SPACE */
      CANCEL          /* PROGRAM CAN BE CANCELLED */
      KEY(5)          /* PROTECT KEY IS 5 */
      SWAP            /* PROGRAM IS SWAPPABLE */
      NOPRIV          /* PROGRAM IS NON PRIVILEGED */
      DSI             /* REQUIRED DATASET INTEGRITY */
      SYST            /* PROGRAM IS A SYSTEM TASK */
      PASS            /* CANNOT BYPASS PASSWORD PROTECTION */
      AFF(NONE)       /* NO CPU AFFINITY */
      NOPREF          /* NO PREFERRED STORAGE FRAMES */

PPT PGMNAME(GDEISBOT) /* DFM SYSTEM INITIALIZATION */
      CANCEL          /* PROGRAM CAN BE CANCELLED */
      KEY(5)          /* PROTECT KEY IS 5 */
      NOSWAP          /* PROGRAM IS NON SWAPPABLE */
      NOPRIV          /* PROGRAM IS NON PRIVILEGED */
      DSI             /* REQUIRED DATASET INTEGRITY */
      SYST            /* PROGRAM IS A SYSTEM TASK */
      PASS            /* CANNOT BYPASS PASSWORD PROTECTION */
      AFF(NONE)       /* NO CPU AFFINITY */
      NOPREF          /* NO PREFERRED STORAGE FRAMES */

PPT PGMNAME(GDEICASB) /* DFM CENTRAL ADDRESS SPACE */
      CANCEL          /* PROGRAM CAN BE CANCELLED */
      KEY(5)          /* PROTECT KEY ASSIGNED IS FIVE */
      NOSWAP          /* PROGRAM IS NON SWAPPABLE */
      NOPRIV          /* PROGRAM IS NON PRIVILEGE */
      DSI             /* REQUIRES DATA SET INTEGRITY */
      SYST            /* PROGRAM IS A SYSTEM TASK */
      PASS            /* CANNOT BYPASS PASSWORD PROTECTION */
      AFF(NONE)       /* NO CPU AFFINITY */
      NOPREF          /* NO PREFERRED STORAGE FRAMES */
```

Figure 61. PPT Entries for Distributed FileManager

Chapter 7. DFMACALL.C sample

The DataAgent sample in [Figure 62 on page 87](#) demonstrates the ability to invoke DFM DataAgent functions from C applications on workstations running SmartData Utilities (SdU) on the DDM client. The sample may need to be modified for your application and your platform. If modified, rename and compile it on a workstation using the header files distributed with SmartData Utilities (SdU). This sample is not included in SYS1.SAMPLIB.

```

/*****
*****  DFMACALL.C  *****/
*****
*
*           DFM DataAgent Sample
*
* Module Name: DFMACALL.C
*
* DDM Workstation Application
*
* Version: 1.0
* Release: 1.0
*
* Copyright (C)
* International Business Machines Corporation 1997
*
* DISCLAIMER OF WARRANTIES: The following (enclosed) code is sample code
* created by the IBM Corporation. This sample code is not a part of any
* IBM product and is provided to you solely for the purpose of assisting
* you in the development of your applications. The code is provided
* "AS IS", without warranty of any kind. IBM shall not be liable for any
* damages arising out of your use of the sample code, even if they have
* been advised of the possibility of such damages.
*
* The sample program does the following:
*
* 1) Construct a filename and filename suffix from the input parameters.
* 2) Do a DDMOpen for the file or directory to trigger MVS suffix
*    processing.
* 3) Do a DDMClose for the file or directory to terminate processing.
*
* COMMAND LINE INVOCATION:
*
* This sample can be invoked in the following formats:
*
* DFMACALL QTSO driveletter:          TSOcommandline [DISPLAY]
* DFMACALL TSO driveletter:          [TSOcommandline] [DISPLAY]
* DFMACALL AGENT driveletter:[filename]
MVSprocedure[,procedural_parameters]
* [PGM program_name] [PARM program-parameters]
[DISPLAY]
* DFMACALL START driveletter:          MVSprocedure[,procedural_parameters]
* DFMACALL driveletter:filename[,filename_suffix] [DISPLAY]
*

```

Figure 62. DFM DataAgent Sample Part 1 of 25

```

* The last format is free-form in which MVS parameters can be specified
* in the filename suffix. Parameters that are relevant to DFM
* DataAgent processing are the following:
* AGENT(agentname) - Specifies the name of a procedure in SYS1.PROCLIB
* that provides the JCL for agent processing and, if PGM is
* omitted, the name of the DataAgent routine (program) to run.
* Note that procedural parameters can also be specified. For
* example, AGENT(agentname,USER=userID,DSNAME=DS1,...). If
* you use the free-form format, remember to specify PARM also.
* PGM(programname) - Specifies the name of the DataAgent routine.
* PARM(program_name) - Specifies input parameters to the DataAgent
* routine.
* START(procedurename,procedure parameters) - Specifies the name of an
* MVS procedure to be started asynchronously.
* DISPLAY - Displays the result of the call to DFM DataAgent. In the case of
* QTSO or TSO the result is the SYSTSPRT file. In the case of other
* DataAgents it is the output name returned by the DataAgent routine
* after successful invocation.
* See DFMXSORT for an example.
* (Note that DISPLAY can be used as a DataAgent name with DFMACALL
* but not as a TSO command.)
*
* Examples:
*
* dfmacall r:ibmuser.a.b
* ==> Opens and closes MVS file ibmuser.a.b on remote drive r.
*
* dfmacall r:ibmuser.a.b,agent(dfmxagnt)
* ==> Opens and closes MVS file ibmuser.a.b invoking agent dfmxagnt.
*
* dfmacall r:ibmuser.a.b,agent(dfmx0001),pgm(dfmxagnt)
* ==> Opens and closes MVS file ibmuser.a.b invoking agent dfmx0001
* ==> with program dfmxagnt.
*
* dfmacall agent r:ibmuser.a.b dfmxagnt
* ==> Opens and closes MVS file ibmuser.a.b invoking agent dfmxagnt
* ==> with default program dfmxagnt and null parameters implied.
*
* dfmacall agent r:ibmuser.a.b dfmxtso pgm ikjeft01
* ==> Opens and closes MVS file ibmuser.a.b invoking agent dfmxtso
* ==> with (APF-authorized) program ikjeft01. (Equivalent to
* ==> DFMACALL TSO.)
*

```

Figure 63. DFM DataAgent Sample Part 2 of 25

```

*      dfmacall start r: dfmx0001,dfminit=iefbr14
*      ==> Asynchronously starts procedure dfmx0001 with parameter
*      ==> dfminit set to iefbr14.
*
*      dfmacall qtso  r: listc  display
*      ==> Calls TSO to list catalog entries and place the results
*      ==> in IBMUSER.DFMQTSO.SYSTSPRT. No SYSTSIN input file is
*      ==> expected. The SYSTSPRT file is displayed.
*
*      dfmacall tso   r:
*      ==> Calls TSO to process input file IBMUSER.DFMXTSO.SYSTSIN
*      ==> and put the results in IBMUSER.DFMXTSO.SYSTSPRT.
*
*      dfmacall tso   r:  "profile prefix(ibmuser)" display
*      ==> After running the command passed it (in this case "profile")
*      ==> it calls TSO to process input file IBMUSER.DFMXTSO.SYSTSIN
*      ==> and put the results in IBMUSER.DFMXTSO.SYSTSPRT. Display the
*      ==> output file.
*
*
*
*****/

#include <os2.h> /* required for VSAM/X applications */
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <malloc.h>
#include "dub.h" /* required master include for VSAM/X applications */

/*-----
--          SYMBOLIC CONSTANTS
-----*/
#define FILCLS_SIZE sizeof(OBJLENGTH) + (2 * sizeof(CODEPOINT))
#define FILCLS_NAME ".DDM_FILCLS"
#define RECDATALEN 100
#define RPYMSBFLN 546 /* reply message buffer length */
#define PATHLEN 300 /* path with 45 extra bytes */
#define PARMLN 255 /* arbitrary maximum parm len */
#define USPARMLN 255 /* arbitrary max user parm len */
#define MINPQTSO 4 /* minimum parameters for QTSO */
#define MAXPQTSO 5 /* maximum parameters for QTSO */
#define MINPTSO 3 /* minimum parameters for TSO */
#define MAXPTSO 5 /* maximum parameters for TSO */
#define MINPAGENT 4 /* minimum parameters for AGENT */
#define MAXPAGENT 9 /* maximum parameters for AGENT */
#define MINPSTART 4 /* minimum parameters for START */
#define MAXPSTART 4 /* maximum parameters for START */
#define MINPFF 2 /* minimum parameters for FF */
#define MAXPFF 3 /* maximum parameters for FF */

```

Figure 64. DFM DataAgent Sample Part 3 of 25

```

/*-----
--          LOCAL FUNCTION DECLARATIONS
-----*/
int SpecialOptions(int index, int argc, char* argv[]);
int CheckRange(int min, int max, int argc, char uarg[PARMLEN]);
VOID DumpBuffer(PDDMOBJECT pAttribute, USHORT Count);
CODEPOINT ReplyMsg(VOID);
VOID OmitError(VOID);
VOID GeneralError(VOID);
VOID ValueError(char *value);
VOID ParmLenError(char *value);
int strupper(char *out, char *in, int buflen);
VOID DisplayBuffer(ULONG count, PDDMRECORD pRcdarea);
VOID DuplicateError(VOID);
VOID HasFileNameError(VOID);
VOID NoFileNameError(VOID);
VOID TooManyError(VOID);
VOID NotEnoughError(VOID);
VOID DisplayHelp(char *helpflag);

/*-----
--          DFMACALL
-----*/

    int dummy_filename = 0;      /* Dummy filename flag          */
    int display_filename = 0;    /* Display filename flag        */
    int TSO_retry = 0;          /* TSO error retry flag         */
    int debug = 1;              /* Debug flag: 0 = nothing displayed, */
                                /* 1 = filename display, 2 = all of */
                                /* the above plus major functions, */
                                /* 5 = all the above plus data.     */
    int display_counter = 0;    /* Record display counter       */
    int intrc;                  /* internal return code         */

main(int argc, char* argv[])
{
    int i;                      /* Loop counter                  */
    int fnlen = 0;              /* Filename length               */
    int pgmcnt = 0;             /* Number of PGM parm occurrences */
    int parmcnt = 0;            /* Number of PARM parm occurrences */
    int data_follows;           /* Data follows in next arg      */
    APIRET SevCode;             /* Severity code (see DUBDEFS.H) */
    CODEPOINT LCodePoint;       /* Local Code Point for reply msg */

```

Figure 65. DFM DataAgent Sample Part 4 of 25

```

PDDMRECORD pRecord;
RECLENGTH RecordSize;
PDDMRECAL pRecAL;
PDDMRECALK pRecALK;
RECLENGTH RecALSize;
PBYTE pData;
HDDMLOAD UnLoad; /* File handle for unload */
ULONG RecCount;
ULONG DDMMoreDataFlag;
int minparms; /* minimum parameters current cmd */
int maxparms; /* maximum parameters current cmd */

HDDMFILE FileHandle;

/* Filename to be operated on */
CHAR MVSFilename[PATHLEN];
CHAR RootName[PATHLEN]; /* Root name for TSO retry */
CHAR dummy_name[9] = "NULLFILE";
CHAR display_name[PATHLEN];

CHAR uarg[PARMLEN]; /* Upper case argument */
CHAR usparg[USPARMLEN]; /* Upper case subparameter */
/*****
/* Determine which command format was used and build the MVS
/* filename and filename suffix accordingly.
*****/

RootName[0] = 0; /* Set TSO root name to null string*/
switch (argc)
{ case 1: /* no user arguments */
    NotEnoughError();
    DisplayHelp("N");
    return(SC_SEVERE);
  default:
    /*****
    /* 1 or more user arguments--check further
    *****/
    /* Convert current user argument to upper case.
    if (intrc = strupper(uarg, argv[1], PATHLEN))
        return intrc;

```

Figure 66. DFM DataAgent Sample Part 5 of 25

```

if (strcmp(uarg,"QTSO") == 0 |
    strcmp(uarg,"TSO") == 0 ) {
if (strcmp(uarg,"QTSO") == 0) {
    /******
    /* QTSO format
    /******
    minparms = MINPQTSO;
    maxparms = MAXPQTSO;
}
else {
    /******
    /* TSO format
    /******
    minparms = MINPTS0;
    maxparms = MAXPTS0;
}
if (intrc = CheckRange(minparms,maxparms,argc, uarg) > 0)
    return(intrc);

    /******
    /* Set special processing flags
    /******
for (i=minparms; i < argc; i++) {
    if (intrc = SpecialOptions(i, argc, argv) > 0)
        return(intrc);
}
    /******
    /* Build filename in the format of
    /* x:fn,agent(dfmqts0,u=userID),pgm(dfmqts0),
    /*      parm(...)
    /*      -- OR --
    /* x:fn,agent(dfmxtso,u=userID),pgm(ikjeft01),
    /*      parm(...)
    /* (Note that MVS will append u=userid)
    /******
if (intrc = strupper(MVSFilename,argv[2],PATHLEN))
    return(intrc);
fnlen = strlen(MVSFilename); /* Save true filename len */
if ( fnlen > 2 | strncmp(&MVSFilename[1],":",1) != 0 ) {
    HasFileNameError();
    DisplayHelp(&uarg[0]);
    return(SC_SEVERE);
}
}

```

Figure 67. DFM DataAgent Sample Part 6 of 25

```

/*****
/* Build root filename.
*/
*****/
if (display_filename) {
    if (strcmp(uarg,"QTS0") == 0)
        strcat(MVSFilename,"DFMQTS0.SYSTSPRT");
    else
        strcat(MVSFilename,"DFMXTS0.SYSTSPRT");
    strcpy(RootName,MVSFilename);
} else {
    dummy_filename = 1;
    strcat(MVSFilename,dummy_name);
} /* endif */

/*****
/* Attach filename suffix.
*/
*****/
if (strcmp(uarg,"QTS0") == 0)
    strcat(MVSFilename,",agent(dfmqts0),parm(");
else
    strcat(MVSFilename,",agent(dfmxts0),pgm(ikjeft01),parm(");

/*****
/* Concatenate parm field
*/
*****/
if (argc >= 4 ) {
    if (intrc = strupper(usparg, argv[3], USPARMLEN))
        return (intrc);
    if (strcmp(usparg,"DISPLAY") != 0)
        strcat(MVSFilename,argv[3]);
    else if (strcmp(uarg,"QTS0") == 0) {
        /* DFMQTS0 requires a parameter field
        */
        OmitError();
        DisplayHelp(&uarg[0]);
        return(SC_SEVERE);
    }
}
strcat(MVSFilename,""); /* Terminate parameter field */
}
/* End of QTS0/TS0 case
*/

```

Figure 68. DFM DataAgent Sample Part 7 of 25

```

else if (strcmp(uarg,"AGENT") == 0) {
/*****
/* AGENT format */
*****/
minparms = MINPAGENT;
maxparms = MAXPAGENT;
if (intrc = CheckRange(minparms,maxparms,argc, uarg) > 0)
    return(intrc);

/*****
/* Set special processing flags */
*****/
for (i=minparms; i < argc; i++) {
    if (intrc = SpecialOptions(i, argc, argv) > 0)
        return(intrc);
}

/*****
/* Build filename in the format of
/* x:fn,agent(agentname),pgm(program_name),parm(parms) */
*****/
if (intrc = strupper(MVSFilename,argv[2], PATHLEN))
    return(intrc);
fnlen = strlen(MVSFilename); /* Save true filename len */
/* Add dummy filename if one wasn't specified */
if ( fnlen == 2 & strncmp(&MVSFilename[1],":",1) == 0 ) {
    dummy_filename = 1;
    strcat(MVSFilename,dummy_name);
}

/* Check the filename format for obvious errors. */
fnlen = strlen(MVSFilename); /* Save true filename len */
if (fnlen < 3 ) {
    NoFileNameError();
    DisplayHelp(&uarg[0]);
    return(SC_SEVERE);
}

strcat(MVSFilename,",agent(");
strcat(MVSFilename,argv[3]);
strcat(MVSFilename,")");
/*****
/* Concatenate optional fields */
*****/
if (argc > 4) {
/*****
/* Optional parameters are present */
*****/

```

Figure 69. DFM DataAgent Sample Part 8 of 25


```

/*****
/* All but DISPLAY are in format of keyword + value. */
/*****
data_follows = 0;
for (i=4; i < argc; i++) {
    if (!data_follows) {
        /* Not data object -- process the keyword. */
        data_follows = 1;
        if (intrc = strupper(usparg, argv[i], USPARMLEN))
            return (intrc);
        if (strcmp(usparg,"PGM") == 0) {
            pgmcnt++;
            strcat(MVSFilename,"pgm(");
        }
        else if (strcmp(usparg,"PARM") == 0) {
            parmcnt++;
            strcat(MVSFilename,"parm(");
        }
        else {
            /* Unidentified keyword?*/
            /* Make sure it's not a display option */
            if (strcmp(usparg,"DISPLAY") != 0 ) {
                GeneralError();
                DisplayHelp(&uarg[0]);
                return(SC_SEVERE);
            }
            else
                data_follows = 0; /* No following data */
        }
        /* End, unidentified kwd*/
    }
    /* End, even number */
    else {
        /* Process the keyword's data. */
        data_follows = 0;
        strcat(MVSFilename,argv[i]);
        strcat(MVSFilename,"");
    }
    /* End, data field */
}
/* End of for loop */
}
/*****
/* Ensure no duplicate parameters */
/*****
if ( pgmcnt > 1 | parmcnt > 1 ) {
    DuplicateError();
    DisplayHelp(&uarg[0]);
    return(SC_SEVERE);
}
/*****
/* Ensure AGENT is invoked with a PARM */
/*****
if (parmcnt == 0)
    strcat(MVSFilename,"parm(");
}
/* End of AGENT */

```

Figure 70. DFM DataAgent Sample Part 9 of 25

```

else if (strcmp(uarg,"START") == 0) {

    /******
    /* START format
    /******
    minparms = MINPSTART;
    maxparms = MAXPSTART;
    if (intrc = CheckRange(minparms,maxparms,argc, uarg) > 0)
        return(intrc);

    /******
    /* Build filename in the format of
    /* x:fn,start(proc,parms)
    /******
    if (intrc = strupper(MVSFilename,argv[2],PATHLEN))
        return(intrc);
    fnlen = strlen(MVSFilename);
    if ( (fnlen > 2) | (strcmp(&MVSFilename[1],":",1) != 0) ) {
        HasFileNameError();
        DisplayHelp(&uarg[0]);
        return(SC_SEVERE);
    }
    dummy_filename = 1;
    strcat(MVSFilename,dummy_name);
    strcat(MVSFilename,"start(");
    strcat(MVSFilename,argv[3]);
    strcat(MVSFilename,")");
}
else {
    /* End of START parameter
    /* None of the above
    /******
    /* Free-form command otherwise
    /******
    /******
    /* Check for help request.
    /******
    /******
    if (strcmp(uarg, "?",1) == 0) {
        DisplayHelp(&uarg[0]);
        return(SC_WARNING);
    }

    minparms = MINPFF;
    maxparms = MAXPFF;
    if (intrc = CheckRange(minparms,maxparms,argc, uarg) > 0)
        return(intrc);
    /******
    /* Set special processing flags
    /******
    for (i=minparms; i < argc; i++) {
        if (intrc = SpecialOptions(i, argc, argv) > 0)
            return(intrc);
    }
}

```

Figure 71. DFM DataAgent Sample Part 10 of 25

```

        if (intrc = strupper(MVSFilename,argv[1], PATHLEN))
            return(intrc);
        fnlen = strlen(MVSFilename); /* Save true filename len */
        if (fnlen < 3) {
            NoFileNameError();
            DisplayHelp(&uarg[0]);
            return(SC_SEVERE);
        }
    }
} /* End of free form parameters */

/*****
/* Begin processing the MVS file
*****/

if (debug >= 1) printf
("DFMACALL: Processing filename and filename suffix of %s.\n",
 MVSFilename);

if (display_filename) {
/*****
/* Perform unload of records to satisfy display request
*****/

/*****
/* Allocate a record buffer
*****/
RecALSize = 64000; /* Try to use a 64K Buffer for records*/
if ((pRecord = (PDDMRECORD)malloc(RecALSize)) == NULL) {
/* Not enough storage--make one last try for 32K
RecALSize = RecALSize / 2; /* Try 32K Buffer
if ((pRecord = (PDDMRECORD)malloc(RecALSize)) == NULL) {
printf("DFMACALL: Out of memory\n");
return(SC_SEVERE);
}
}
}

/*****
/* Unload the first batch of records in rcd number order
*****/
SevCode = 1;
while (SevCode) {
    SevCode = DDMUnLoadFileFirst
        (MVSFilename, /* FileName
        &UnLoad, /* UnLoadHandle
        0UL, /* AccessFlags
        &DDMMoreDataFlag, /* Flags
        pRecord, /* RecordBuf
        RecALSize, /* RecordBufLen
        (CODEPOINT) RECSEQ, /* UnloadOrder=rcd number
        &RecCount /* RecCount
    );
}

```

Figure 72. DFM DataAgent Sample Part 11 of 25

```

    if (SevCode > SC_WARNING) {
        printf("DFMACALL: Error on DDMUnLoadFileFirst for %s.\n",
            MVSFilename);
        printf("Severity code = %u\n",SevCode);
        LCodePoint = ReplyMsg();
        /* Retry TSO unload output file if not tried already.      */
        if (strcmp(MVSFilename,RootName) != 0 &
            LCodePoint == VALNSPRM ) {
            TSO_retry = 1;
            strcpy(MVSFilename,RootName);
        }
        else {
            /* End, TSO retry      */
            /* Permanent error      */
            free(pRecord);
            return(SevCode);
        }
        /* End, permanent error */
    }
    /* End of error from unload file first. */
}
/* End of while no error. */

if (SevCode > SC_WARNING) {
    free(pRecord);
    return(SevCode);
}
/* End, unload first err */

if (debug >= 2 )
    printf ("DDMUnLoadFileFirst: %d records in buffer .\n",
        RecCount);
if (TSO_retry > 0 ) {
    printf ("\n** The TSO Output File associated with the error is\
as follows: **\n");
    printf ("**      (Note that its contents may be from a previou\
s run.) **\n");
}

DisplayBuffer(RecCount,pRecord);

/*****
/* Unload remaining records in record number order.      */
/* When DDMMoreDataFlag is 0x00UL then the file handle is  */
/* invalid and the file will be closed.                    */
*****/
while (DDMMoreDataFlag == 0x01UL)
{
    SevCode = DDMUnLoadFileNext
        (UnLoad,
         0x0000UL,
         &DDMMoreDataFlag,
         pRecord,
         RecALSize,
         &RecCount
        );
    /* UnLoadHandle      */
    /* Flags            */
    /* UnloadFlags      */
    /* RecordBuf        */
    /* RecordBufLen     */
    /* RecCount         */
}

```

Figure 73. DFM DataAgent Sample Part 12 of 25

```

        if (SevCode > SC_WARNING) {
            printf("DFMACALL: Error on DDMUnLoadFileNext for %s.\n",
                MVSFilename);
            printf("Severity code = %u\n",SevCode);
            ReplyMsg();
            free(pRecord);
            return(SevCode);
        }
        if (debug >= 2)
            printf ("DDMUnLoadFileNext: %d records in buffer.\n",
                RecCount);
        DisplayBuffer(RecCount,pRecord);
    }
    free(pRecord);                                     /* End of WHILE loop */

} else {
    /******
    /* No display--just open to trigger DataAgent and then close */
    /******

    /******
    /* Open the file */
    /******
    SevCode = DDMOpen
        (MVSFilename,                                /* FileName      */
         &FileHandle,                                /* FileHandle    */
         RELRNBAM,                                    /* AccessMethod  */
         DDM_GETAI,                                    /* AccIntList    */
         DDM_UPDATERS,                                /* FileShare     */
         NULL,                                         /* EABuf         */
         NULL,                                         /* reserved      */
         );
    if (SevCode != SC_NO_ERROR)
    {
        if (dummy_filename == 0)
        {
            printf("Error opening file %s\n",MVSFilename);
            printf("Severity code = %u\n",SevCode);
        }
        ReplyMsg();
        return(SevCode);
    }
}

```

Figure 74. DFM DataAgent Sample Part 13 of 25

```

/*****
/* Close the file */
*****/
SevCode = DDMClose
          (FileHandle          /* FileHandle */
          );

if (SevCode != SC_NO_ERROR)
{
    if (dummy_filename == 0)
    {
        printf("Error closing file %s\n",MVSFilename);
        printf("Severity code = %u\n",SevCode);
    }
    ReplyMsg();
    return(SevCode);
}
}

return(SC_NO_ERROR);

} /* End--sample main */

/*****
***** ReplyMsg *****
*****
* Process the reply message if there is a Severity Code other than
* SC_NO_ERROR;
*
*****/
CODEPOINT ReplyMsg(VOID)
{
    static BYTE pRpyMsgBuf[RPYMSBFLN];

    APIRET rc;
    CODEPOINT CodePoint;
    PDDMOBJECT pReplyObject;
    USHORT index;

```

Figure 75. DFM DataAgent Sample Part 14 of 25


```

7, "Record Number Out of Bounds      \0",
5, "Record Not Found                 \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
3, "Invalid Key Length                \0",
0, "                                 \0",
0, "                                 \0",
3, "Not Authorized to Access Method  \0",
0, "Invalid Access Method             \0",
3, "Permanent Agent Error            \0",
6, "Resource Limits Reached on Target System \0",
3, "Invalid Base File Name           \0",
0, "                                 \0",
0, "                                 \0",
2, "Not Authorized to Directory      \0",
0, "Management Class Conflict         \0",
0, "Storage Class Conflict            \0",
3, "Existing Condition               \0",
4, "Not Authorized to File           \0",
6, "Invalid Request                  \0",
4, "Invalid Key Definition            \0",
0, "                                 \0",
5, "Key Update Not Allowed by Same Index \0",
8, "Invalid Key Value                \0",
0, "                                 \0",
0, "                                 \0",
3, "Open Exclusive by Same User      \0",
4, "Concurrent Open Exceeds Maximum  \0",
4, "Conversational Protocol Error    \0",
0, "                                 \0",
0, "                                 \0",
0, "                                 \0",
7, "Record Damaged                   \0",
7, "Record in Use                    \0",
0, "                                 \0",
5, "Data Stream Syntax Error         \0",
7, "Update Cursor Error              \0",
5, "No Update Intent on Record       \0",
3, "Invalid New File Name            \0",
3, "Function Not Supported            \0",
3, "Parameter Not Supported           \0",
4, "Parameter Value Not Supported     \0",
4, "Object Not Supported              \0",
5, "Command Check                    \0",
0, "                                 \0",

```

Figure 77. DFM DataAgent Sample Part 16 of 25


```

0, "                                \0",
2, "File Handle Not Found          \0",
3, "Directory Full                  \0",
3, "Record Inactive                 \0",
7, "File Damaged                    \0",
4, "Load Records Count Mismatch     \0",
3, "Not Authorized to Open Intent for Named File \0",
0, "                                \0",
3, "File Closed with Damage          \0",
2, "Target Not Supported             \0",
5, "Key Value Modified after Cursor was Last Set \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "Access Intent List Error         \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
5, "Record Not Available            \0",

/***** START OF SECOND CODE POINT RANGE *****/
0, "OS/2 Error                      \0",
0, "Data Description File Not Found  \0",
0, "Conversion Table Not Found        \0",
2, "Translation Error                 \0",
0, "                                \0",
2, "Invalid Flag                      \0",
0, "                                \0",
2, "Communications Error              \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
0, "                                \0",
2, "Resource Limit Reached in OS/2 V2.0 Source System \0",
2, "Field Length Error                \0",
2, "Address Error                     \0",
0, "                                \0",
2, "Function Continuation Error       \0",
0, "                                \0",
2, "File Error                        \0",
};

```

Figure 78. DFM DataAgent Sample Part 17 of 25

```

/*-----
-- For each reply message available, retrieve and display it.
-----*/
do
{ /*-----
-- Get the reply message
-----*/
rc = DDMGetReplyMessage(pRpyMsgBuf, (ULONG)RPYMSBFLN, (ULONG)1);

switch (rc)
{ case SC_NO_ERROR: /* All reply messages have been received */
  case SC_WARNING: /* There are more reply messages to be received*/
    break;
  case SC_ERROR:
    printf(" ReplyMsg: reply message buffer is too small -\n");
    printf("          enlarge and recompile ...\n");
    return(rc);
    break;
  case SC_SEVERE:
    printf(" ReplyMsg: Warning: A reply message was requested,\n");
    printf("          but there are none available ...\n");
    return(rc);
    break;
  case SC_ACCESSDAMAGE:
    printf(" ReplyMsg: Error: An invalid reply message buffer\n");
    printf("          address was specified ...\n");
    return(rc);
    break;
  case SC_PERMDAMAGE:
    printf(" ReplyMsg: Severe Error: An unarchitected reply message\n");
    printf("          object was encountered ...\n");
    return(rc);
    break;
  default:
    printf(" ReplyMsg: Unknown return code from DDMGetReplyMessage\n");
    return(rc);
    break;
} /* endswitch */

```

Figure 79. DFM DataAgent Sample Part 18 of 25

```

/*-----
-- Get the reply message
-----*/
pReplyObject = (PDDMOBJECT)pRpyMsgBuf;

CodePoint = pReplyObject->cpObject;          /* get code point */

/* reset pointer to first parm base */
pReplyObject = (PDDMOBJECT)((PBYTE)pReplyObject
                             + (sizeof(CODEPOINT)
                               + sizeof(OBJLENGTH))
                             );

/*-----
-- Calculate the index into the parameter/msg table based on
-- the codepoint.
-----*/
if (CodePoint <= RECNAVRM)          /* if code point in first block */
    index = (USHORT)(CodePoint - KEYUDIRM);
else                                /* code point in second block */
    index = (USHORT)
        ((RECNAVRM - KEYUDIRM + 1) /* number of entries in
                                     first block */
         + (CodePoint % 0x0100UL) /* index into second block */
        );

/* If the index indicates "file not found" and a dummy filename */
/* is being used, ignore the error. */
if (index == 13 & dummy_filename == 1 )
    return(SC_NO_ERROR);

/*-----
-- Begin dissecting the reply message buffer
-----*/
if (ErrorMsgBuffer[index].Count > 0)
{
    printf("RPYMSG: %s\n",ErrorMsgBuffer[index].msg);
    DumpBuffer(pReplyObject, ErrorMsgBuffer[index].Count);
    printf("\n");
}

} while (rc == SC_WARNING); /* enddo */
return(CodePoint);
} /* ReplyMsg */

```

Figure 80. DFM DataAgent Sample Part 19 of 25

```

/*****
***** DumpBuffer *****/
*****
*   For each object in the reply message buffer, print out its contents.
*
*****/
VOID DumpBuffer(PDDMOBJECT  pAttribute,
               USHORT      Count)
{
    int i;                                /* Local loop counter */
    do
    { if (pAttribute->cbObject == (sizeof(CODEPOINT) + sizeof(OBJLENGTH)))
      { printf("Null object returned = %x\n", pAttribute->cbObject);
        pAttribute->cpObject = 0;
      }
      else
      { switch(pAttribute->cpObject)
        { case ACCMTHCL:                /* Access Method Class */
          printf("ACCMTHCL = 0x%X\n", *(PCODEPOINT)(pAttribute->
>pData));
          break;
          case BASFILNM:                /* Base File Name */
          printf("BASFILNM = %s\n", pAttribute->pData);
          break;
          case CODPNT:                 /* Code Point */
          printf("CODPNT = 0x%X\n", *(PCODEPOINT)(pAttribute->
>pData));
          break;
          case CSRPOSST:               /* Cursor Position Status */
          printf("CSRPOSST = 0x%hX\n", *(PBYTE)(pAttribute->
>pData));
          break;
          case DTALCKST:               /* Data Lock Status */
          printf("DTALCKST = 0x%hX\n", *(PBYTE)(pAttribute->
>pData));
          break;
          case ERRFILNM:               /* Error File Name */
          printf("ERRFILNM = %s\n", pAttribute->pData);
          break;
          case FILNAM:                 /* File Name */
          printf("FILNAM = %s\n", pAttribute->pData);
          break;
          case KEYDEFCD:               /* Key Definition Error Code */
          printf("KEYDEFCD = 0x%hX\n", *(PBYTE)(pAttribute->
>pData));
          break;
          case MAXOPN:                 /* Maximum Number of File Extents
                                     Concurrent Opens Allowed */
          printf("MAXOPN = %d\n", *(PUSHORT)(pAttribute->pData));
          break;
        }
      }
    } while (Count--);
}

```

Figure 81. DFM DataAgent Sample Part 20 of 25

```

        case NEWFILNM:                /* New File Name */
            printf("NEWFILNM = %s\n", pAttribute->pData);
            break;
        case PRCCNVCD:                /* Conversational Protocol Error Code */
            printf("PRCCNVCD = 0x%hX\n", *(PBYTE)(pAttribute->pData));
            break;
        case RECCNT:                  /* Record Count */
            printf("RECCNT = %ld\n", *(PULONG)(pAttribute->pData));
            break;
        case RECNR:                   /* Record Number */
            printf("RECNR = %ld\n", *(PRECNUM)(pAttribute->pData));
            break;
        case SRVDGN:                  { /* Server Diagnostic Information */
            printf("SRVDGN = 0x\n");
            for (i=1; i < (pAttribute->cbObject-5); i++) /* 2 byte len, 2 byte codept*/
                if (i % 16 ==0)
                    printf("%02X\n", *(PBYTE)(pAttribute->pData+i-1));
            else
                if (i % 4 ==0)
                    printf("%02X ", *(PBYTE)(pAttribute->pData+i-1));
            else
                printf("%02X", *(PBYTE)(pAttribute->pData+i-1));
            }
            break;
        case SVRCOD:                  /* Severity Code */
            printf("SVRCOD = 0x%X\n", *(PCODEPOINT)(pAttribute->pData));
            break;
        case SYNERRCD:                /* Syntax Error Code */
            printf("SYNERRCD = 0x%hX\n", *(PBYTE)(pAttribute->pData));
            break;
        default:
            printf("Unknown code point - 0x%X\n", *(PCODEPOINT)(pAttribute->pData));
            break;
    } /* endswitch */
} /* endif */

/* go to next object */
pAttribute = (PDDMOBJECT)((PBYTE)pAttribute + pAttribute->cbObject);

} while(--Count > 0);

} /* DumpBuffer */

```

Figure 82. DFM DataAgent Sample Part 21 of 25

```

/*****
***** Error Routines *****/
/*****
VOID GeneralError()
{
    printf("DFMACALL: Incorrect command line syntax.\n");
} /* GeneralError */

VOID OmitError()
{
    printf("DFMACALL: A required parameter was omitted.\n");
}

VOID TooManyError()
{
    printf("DFMACALL: Too many parameters were on the command line.\n");
}

VOID NotEnoughError()
{
    printf("DFMACALL: Not enough parameters were on the command line.\n");
}

VOID HasFileNameError()
{
    printf("DFMACALL: Filename is not allowed for QTSO, TSO, or START.\n");
}

VOID NoFileNameError()
{
    printf("DFMACALL: A filename must be specified.\n");
}

VOID ValueError(char *value)
{
    printf("DFMACALL: Incorrect parameter value %s.\n",value);
}

VOID ParmLenError(char *value)
{
    printf("DFMACALL: Parameter %s is too long.\n",value);
}

int strupper(char *oarg, char *iarg, int buflen)
{
    /* Convert string to upper case. */
    int i;
    if (strlen(iarg) > buflen) {
        ParmLenError(iarg);
        return(SC_SEVERE);
    }
}

```

Figure 83. DFM DataAgent Sample Part 22 of 25

```

    for (i=0; i < strlen(iarg); i++) {
        oarg[i] = toupper(iarg[i]);
    }
    oarg[i] = 0;
    return (0);
}

VOID DuplicateError()
{
    printf("DFMACALL: One or more parameters were duplicated.\n");
}

VOID DisplayBuffer(ULONG count, PDDMRECORD pCurrentRecord)
{
    /* Display a buffer full of records */
    ULONG i; /* record counter */
    int j; /* index to character in record */
    int cRecLen; /* current record length */
    UCHAR c; /* current converted character */
    UCHAR savechar; /* savearea for trailing character */

    for (i=1; i <= count; i++) {
        cRecLen = pCurrentRecord->cbRecord - sizeof(pCurrentRecord->cbRecord)
            - sizeof(pCurrentRecord->cpRecord);
        /******
        /* Replace all instances of non-printable characters,
        /******
        /* Make sure the string is printable and
        /* make sure that there is no 0 in the middle of string.
        for (j=0; j < cRecLen; j++) {
            if (!(c = pCurrentRecord->pRecord[j]))
                pCurrentRecord->pRecord[j] = ' '; /* Replace x00 with blank */
            else if (!isprint(c))
                pCurrentRecord->pRecord[j] = '.'; /* Make nonprintable a "." */
        } /* End of for j= loop */
        savechar = pCurrentRecord->pRecord[cRecLen]; /* save trailing char */
        pCurrentRecord->pRecord[cRecLen] = '\0';

        if (debug >= 5) {
            display_counter++;
            printf ("Displaying record %d with length %d:\n",
                display_counter, cRecLen);
        }
        printf ("%s\n", pCurrentRecord->pRecord);
        pCurrentRecord->pRecord[cRecLen] = savechar; /* restore trailing */
        pCurrentRecord = (PDDMRECORD) (pCurrentRecord->pRecord + cRecLen);
    }
}

```

Figure 84. DFM DataAgent Sample Part 23 of 25

```

    SpecialOptions(int index, int argc, char *argv[])
{
    CHAR uarg[PARMLEN];          /* Upper case argument          */
    /* Check for special processing options.                      */
    if (intrc = strupper(uarg, argv[index], PARMLEN))
        return(intrc);
    if (strcmp(uarg,"DISPLAY") == 0)
        display_filename = 1;

    return(0);
} /* End of SpecialOptions */

    CheckRange(int minparms, int maxparms, int argc, char uarg[PARMLEN])
{
    /* Ensure number of parameters is reasonable for the command */

    /******
    /* Ensure enough parameters
    /******
    if (argc < minparms) {
        NotEnoughError();
        DisplayHelp(&uarg[0]);
        return(SC_SEVERE);
    }

    /******
    /* Ensure no leftover parameters
    /******
    if (argc > maxparms) {
        TooManyError();
        DisplayHelp(&uarg[0]);
        return(SC_SEVERE);
    }

    return(0);
}

```

Figure 85. DFM DataAgent Sample Part 24 of 25


```

/*****
***** DisplayHelp *****
*****
*   Display the correct syntax for invoking this function.
*
*****/
VOID DisplayHelp(char *fullhelp)
{
if (strncmp(fullhelp,"?",1) == 0) {
/* Print full help text.
printf("Correct syntax:
printf(" DFMACALL QTSO driveletter: TS0commandline [DISPLAY]
printf(" DFMACALL TSO driveletter: [TS0commandline] [DISPLAY]
\n");
printf(" DFMACALL AGENT driveletter:[filename]
MVSproc[,proc_parms] \n");
printf(" [PGM prog_name] [PARM prog_parms]
[DISPLAY] \n");
printf(" DFMACALL START driveletter: MVSproc[,proc_parms] \n");
printf(" DFMACALL driveletter:filename[,filename_suffix]
[DISPLAY] \n\n");
printf(" Examples:
printf(" dfmacall qtso r: listc display
printf(" dfmacall tso r: display
printf(" dfmacall agent r:ibmuser.a.b dfmxagnt
printf(" dfmacall agent r:ibmuser.a.b dfmxtso
printf(" pgm ikjeft01 parm listc
printf(" dfmacall start r: dfmx0001,dfminit=iefbr14
printf(" dfmacall r:ibmuser.a.b,agent(dfmxagnt),parm(hello)
printf(" dfmacall r:ibmuser.a.b,agent(dfmx0001),pgm(dfmxagnt)\n\n");
}
else {
/* Print clue for getting correct help text.
printf("DFMACALL: Enter DFMACALL ? to get the correct command syntax. \n\n");
}
}
}

```

Figure 86. DFM DataAgent Sample Part 25 of 25

Chapter 8. DDM File Attributes

Table 2 on page 113 summarizes the DDM file attributes by DDM file class.

Table 2. DDM File Attributes

Attribute	Description	Chg	PDSE Member (SEQFIL)	PDSE Member (STRFIL)	SAM	ESD S	KSDS/ VRRDS	RRDS
ACCMTHLS	Access method list	NO	YES	YES	YES	YES	YES	YES
DELCF	Delete capable	NO	YES	...	YES	YES	YES	YES
DFTREC	Default record	NO	YES	...	NO	NO	NO	NO
DTACLSNM	Data class name	NO	YES	YES	YES	YES	YES	YES
DTAFMT	Data format	YES	...	YES
EOFNBR	End of file number	NO	YES	YES	NO	YES	YES	YES
EOFF	End of file offset	NO	...	YES
FILEBYTCN	File byte count	NO	YES	YES	NO	NO	NO	NO
FILCHGDT	File change date	YES	YES	YES	YES	YES	YES	YES
FILCLS	File class	NO	YES	YES	YES	YES	YES	YES
FILCRTDT	File creation date	NO	YES	YES	YES	YES	YES	YES
FILEXNCN	File extent count	NO	NO	NO	NO	NO	NO	NO
FILEXNSZ	File extent size	NO	YES	YES	NO	NO	NO	YES
FILEXPDT	File expiration date	YES	YES	YES	YES	YES	YES	YES
FILHDD	File hidden	ME M	YES	YES	YES	YES	YES	YES
FILINISZ	Initial file size	NO	YES	YES	NO	NO	NO	YES
FILMAXEX	Maximum number extents	NO	YES	YES	YES	YES	YES	YES
FILNAM	File name	NO	YES	YES	YES	YES	YES	YES
FILOPNLO	Open lock options	NO	YES	NO	NO	NO	NO	...
FILPRT	File protected	YES	YES	YES	YES	YES	YES	YES
FILSIZ	File size	NO	YES	...	NO	YES
FILSYS	System file	ME M	YES	YES	YES	YES	YES	YES
GETCF	Get capable	ME M	YES	YES	YES	YES	YES	YES
INSCP	Insert capable	ME M	YES	...	YES	YES	YES	YES
KEYDEF	Key definition	NO	YES/

Table 2. DDM File Attributes (continued)

Attribute	Description	Chg	PDSE Member (SEQFIL)	PDSE Member (STRFIL)	SAM	ESD S	KSDS/ VRRDS	RRDS
KEYDUPCP	Duplicate key capable	NO	YES/
LSTACCDT	Last access date	NO	NO	NO	YES	YES	YES	YES
LSTARCDT	Last archive date	NO	YES	YES	NO	NO	NO	NO
MGMCLSNM	Management class name	DS	YES	YES	YES	YES	YES	YES
MODCP	Modify capable	MEM	YES	YES	YES	YES	YES	YES
RECLEN	Record length	NO	YES	...	YES	YES	YES	YES
RECLENCL	Record length class	NO	YES	...	YES	YES	YES	YES
RTNCLS	Retention class	NO	YES	YES	YES	YES	YES	YES
SHDEXS	Shadow exists	NO	YES	YES	YES	YES	YES	YES
STGCLSNM	Storage class name	DS	YES	YES	YES	YES	YES	YES
STRSIZ	Stream size	YES	...	NO
TITLE	Title	MEM	YES	YES	NO	NO	NO	NO

Legend for Chg column:

YES

Attribute value can be changed with the CHGFAT command for a full access data set.

NO

Attribute value cannot be changed with the CHGFAT command.

DS

Attribute value can be changed for data sets only with the CHGFAT command.

MEM

Attribute value can be changed for PDSE members only with the CHGFAT command.

Legend for Data Set/Member columns:

YES

Supported; that is, an attribute value is returned when requested on a LSTFAT command.

NO

Not supported; that is, no attribute value is returned when requested on a LSTFAT command.

...

Does not apply to any of the possible file classes for this data set member.

Chapter 9. Application Programming Considerations

This appendix contains programming considerations relevant to a DDM client record and stream file access. When programming with these application programming interfaces (APIs), you need to consider Distributed FileManager file creation support, access command support, access restrictions, and logon mode requirements.

Distributed FileManager Implementation

DFM provides a subset of DDM access methods, file types, and commands. In some cases, DFM does not support certain record access and stream file API commands or command parameters.

DDM Record Access File Creation

DFM creates record-oriented files based on DDMCreateRecFile command parameter settings. The following are optional parameter settings which govern data set creation within the indicated FileClass. Also provided are some mandatory flag and parameter settings for supported DFM functions.

FileClass SEQFIL

Results in the creation of a SAM data set, a PDSE member, a PDS member, or a VSAM RRDS or VRRDS.

- The type of data set created depends on these parameters:
 - If the FileName parameter includes a *member name*, and the Delete Capability parameter is *off*, a PDSE member or PDS member is created. (If the data set does not exist, a PDSE is created first, and then the member is created.)
 - If the FileName does *not* include a member name, and Delete Capability is *on*, a VSAM RRDS or VRRDS is created.
 - If the FileName does *not* include a member name, and Delete Capability is *off*, a SAM data set is created.
- FileClass SEQFIL CreateFlags mandatory settings are:

Set off the following bit flag: DDM_TMPFIL

FileClass KEYFIL

Results in the creation of a VSAM KSDS data set.

- FileClass KEYFIL CreateFlags mandatory settings are:

Set off the following bit flags: DDM_TMPFIL, DDM_ALDUPKEY
- In FileClass KEYFIL, for the parameters DftRec and DftRecOp, the only valid value is NIL.

FileClass DIRFIL

Results in the creation of a VSAM RRDS or VRRDS data set.

- The type of data set created is determined by the following:
 - A RecLenCls value of RECFIX results in a RRDS data set
 - A RecLenCls value of RECIVL or RECVAR results in a VRRDS data set
- FileClass DIRFIL CreateFlags mandatory settings are:

Set off the following bit flag: DDM_TMPFIL

Additional Considerations

You should also be aware that:

- If you do not specify InitFileSiz for DDMCreateRecFile, the file size is determined by your ACS routines.

Stream File Creation

You can use the DDM Stream access method to create stream files in SAM data sets or PDSE members. The type of data set created depends on:

- If the FileName parameter does not include a member name, a SAM data set is created.
- If the FileName includes a member name, a PDSE member is created. (If the data set does not exist, a PDSE is created first, and then the member is created.)

You should also be aware that:

- DFM uses a tunable parameter in DFM00 called STREAM_LRECL for record length and RECFM = V to create new SAM data sets or PDSE data sets. These attributes override LRECL and RECFM of the SMS data class defaulted by the ACS routine.

File Access Commands Supported by Distributed FileManager

Only commands supported by the following DDM access methods can be issued using DFM:

Access Method Description

RELNRBAM

Relative by record number access method

RNDRNBAM

Random by record number access method

CMBRNBAM

Combined record number access method

RELKEYAM

Relative by key access method

RNDKEYAM

Random by key access method

CMBKEYAM

Combined key access method

STRAM

Stream access method

Sequential Files

Table 3 on page 116 lists the DDM access method commands supported for DFM sequential files.

Table 3. DDM Access Method Commands Supported for Distributed FileManager Sequential Files

Access Commands	RELNRBAM	RNDRNBAM	CMBRNBAM	STRAM
CHGEOF	YES*
CLOSE	YES	YES	YES	YES
DELREC	YES#	YES#	YES#	...
FRCBFF	YES	YES	YES	YES
GETREC	YES	YES	YES	...
GETSTR	YES

Table 3. DDM Access Method Commands Supported for Distributed FileManager Sequential Files
(continued)

Access Commands	RELNRBAM	RNDRNBAM	CMBRNBAM	STRAM
INSRECEF	YES	YES	YES	...
INSRECNB	...	YES+	YES+	...
LCKSTR	YES
MODREC	YES	YES	YES	...
OPEN	YES	YES	YES	YES
PUTSTR	YES*
SETBOF	YES	YES	YES	...
SETEOF	YES@	YES@	YES@	...
SETFRS	YES	YES	YES	...
SETLST	YES@	YES@	YES@	...
YES	...	YES	...	
...	YES	YES	...	
SETNXT	YES	...	YES	...
YES	...	YES	...	
YES	...	YES	...	
NO	YES	YES	...	
UNLIMPLK	YES	YES	YES	...
UNLSTR	Yes

Legend:

YES

The command is supported.

NO

The command is not supported.

...

The command does not apply to the access method.

#

DELREC is only supported for RRDSs and VRRDSs.

@

SETEOF and SETLST are not supported for PDS members.

+

INSRECNB returns “duplicate record number” for PDSE members and PDS members.

*

DFM limits stream access to read-only support for VRRDS and RRDS data sets. All stream access to non-reusable VSAM data sets is read-only.

Direct Files

Table 4 on page 118 lists the DDM access method commands supported for DFM direct files.

Table 4. DDM Access Method Commands Supported for Distributed FileManager Direct Files

Access Commands	RELRNBAM	RNDRNBAM	CMBRNBAM	STRAM
CHGEOF	YES*
CLOSE	YES	YES	YES	YES
DELREC	YES	YES	YES	...
FRCBFF	YES	YES	YES	YES
GETREC	YES	YES	YES	...
GETSTR	YES
INSRECEF	YES	YES	YES	...
INSRECNB	...	YES	YES	...
LCKSTR	YES
MODREC	YES	YES	YES	...
OPEN	YES	YES	YES	YES
PUTSTR	YES*
SETBOF	YES	YES	YES	...
SETEOF	YES	YES	YES	...
SETFRS	YES	YES	YES	...
SETLST	YES	YES	YES	...
YES	...	YES	...	
...	YES	YES	...	
SETNXT	YES	...	YES	...
YES	...	YES	...	
YES	...	YES	...	
NO	YES	YES	...	
YES	YES	YES	...	
UNLSTR	YES

Legend:

YES

The command is supported.

NO

The command is not supported.

...

The command does not apply to the access method.

DFM limits stream access to read-only support for RRDSs and VRRDSs. All stream access to non-reusable VSAM data sets is read-only.

Keyed Files

Table 5 on page 119 lists the DDM access method commands supported for DFM keyed files.

Table 5. DDM Access Method Commands Supported for Distributed FileManager Keyed Files

Access Commands	RELKEYAM	RNDKEYAM	CMBKEYAM	STRAM
CHGEOF	YES*
CLOSE	YES	YES	YES	YES
DELREC	YES#	YES#	YES#	...
FRCBFF	YES	YES	YES	YES
GETREC	YES	YES	YES	...
GETSTR	YES
INSRECKY	YES	YES	YES	...
INSRECNB	NO	NO	NO	...
LCKSTR	YES
MODREC	YES	YES	YES	...
OPEN	YES	YES	YES	YES
PUTSTR	YES*
SETBOF	YES	YES	YES	...
SETEOF	YES	YES	YES	...
SETKEY	YES@	YES	YES	...
SETKEYFR	YES	YES	YES	...
SETKEYLM	YES	...	YES	...
SETKEYLS	YES	YES	YES	...
SETKEYNX	YES	...	YES	...
SETKEYPR	YES	...	YES	...
SETNXTKE	YES	...	YES	...
YES	YES	YES	...	
YES	YES	YES	...	
UNLSTR	YES

Legend:

Yes

The command is supported.

No

The command is not supported.

...

The command does not apply to the access method.

*

DFM limits stream access to read-only support for keyed files.

#

DELREC is only supported for KSDSs.

@

RELKEYAM is promoted to CMBKEYAM.

Stream Files

Table 6 on page 120 lists the DDM access method commands supported for DFM stream files.

Table 6. DDM Access Method Commands Supported for Distributed FileManager Stream Files

Access Commands	STRAM
CHGEOF	YES
CLOSE	YES
FRCBFF	YES
GETSTR	YES
LCKSTR	YES
OPEN	YES
PUTSTR	YES
UNLSTR	YES
Legend:	
YES The command is supported.	

DDM Record Access Restrictions

Restrictions for applications doing record access to DFM are:

- DFM does not support the following functions for accessing multivolume data sets:
 - Backward processing functions: DDMSetMinus and DDMSetPrevious
 - Direct positioning functions: DDMSetBOF, DDMSetEOF, DDMSetFirst, and DDMSetLast
 - DDMInsertRecEOF function (An alternative is to use the DDMLoadFileFirst function to write records to an empty file or to extend the file.)
 - Under some conditions, DDMGetRec and DDMMModifyRec functions for accessing records that span physical volumes

The equivalent DDM commands for these DDM record access functions are SETPRV, SETMNS, SETBOF, SETEOF, SETFRS, SETLST, INSRECEP GETREC and MODREC. The TRGNSPRM reply message is returned if these commands are used to access multivolume data sets.

- The DDMOpen AccIntList includes DDM_MODAI, DDM_INSAI, and DDM_GETAI bit flags. You must explicitly state all your access intents for the duration of the file being open under control of that DDMOpen command.

There is not a more powerful access intent which implicitly permits another less powerful access. For example, if you specify DDM_MODAI but not DDM_GETAI and then attempt to retrieve a record, you will receive an error reply message.

- The following are other DFM access restrictions:
 - For the DDMMModifyRec command, the AccessFlags DDM_INHMODKY bit flag must be set on.
 - For DDMUnLoadFileFirst, you must explicitly specify UnloadOrder KEYORD for KSDS.
 - For DDMSetKey, the AccessFlag DDM_HLDCSR bit flag must be set off.
 - DDMSetLast with the AccessFlag DDM_RECNRFB bit flag set on for ESDS and partitioned sequential data sets returns the special value of -1 for the record number feedback. This indicates that the number is not known.

- If you are accessing KSDSs or AIFs, you can only use DDM keyed access method commands. You can only access these records by key, not by record number.
- If you are accessing a PDS member:
 - Only sequential load (using INSRECEP or LODRECF) is supported; random load (by record number) is not supported.
 - These DDM access commands are not supported: DELREC, INRECNB, SETEOF, and SETLST.

Stream File API Restrictions

Stream file API restrictions, from DDM clients:

A DDM client editor might report a critical error when you create a remote stream file using DFM. The file is, however, actually created and can subsequently be accessed without editor error messages.

Logon Mode Requirements

The IBM i5/OS system and DDM clients have the following logon mode requirements:

- It requires a logon mode named QPCSUPP in order to perform authorization checking.
- It only supports one logon mode name specification to be used for all target systems.
- If you do not successfully place a logon mode table entry named QPCSUPP in the z/OS VTAM logon mode table concatenation for your z/OS APPC APPL, you will receive error messages, see [Figure 87 on page 121](#), on your z/OS console:

```
IST663I BFINIT REQUEST FROM DFMNCP (my NCP major node) FAILED,
SENSE=...
IST664I REAL OLU=PELNET01.PS2ILU1      REAL DLU=PELNET01.DFMILU1
IST889I SID=...
IST264I REQUIRED LOGMODE NAME QPCSUPP UNDEFINED
IST314I END
IST663I BFTerm REQUEST FROM DFMNCP RECEIVED, SENSE=...
IST664I (same as before)
IST889I SID=...
IST891I PELNET01.VTAMF GENERATED FAILURE NOTIFICATION
IST893I ORIGINAL FAILING REQUEST IS BIND
IST314I END
```

Figure 87. Error Messages

Appendix A. Accessibility

Accessible publications for this product are offered through [IBM Documentation \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the [Contact the z/OS team web page \(www.ibm.com/systems/campaignmail/z/zos/contact_z\)](http://www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Documentation with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1)

are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element

The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element

The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE (KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. 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. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

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

*IBM Corporation
Site Counsel
2455 South Road*

Poughkeepsie, NY 12601-5400
USA

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This publication documents intended Programming Interface that allow the customer to write programs to obtain services of DFSMS.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux® is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in DFSMSHsm documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMSHsm manual.

This glossary includes terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

access method

(1) A mainframe data management routine that moves data between storage and an I/O device in response to requests made by a program. (2) The part of the distributed data management architecture which accepts commands to access and process the records of a file.

ACS

See *Automatic class selection (ACS)*.

Advanced Program-to-Program Communications (APPC)

An implementation of the Systems Network Architecture (SNA) logical unit (LU) 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

agent

Manages the parsing and routing of DDM commands and replies.

AIF

See *Alternate index file*.

AIX

Advanced Interactive Executive

alias

An alternative name for an ICF user catalog, a non-VSAM file, or a member of a partitioned data set (PDS) or PDSE.

alternate index file

A file that supports keyed forms of access to the records of a base file.

API

See *application programming interface (API)*.

APPC

See *Advanced Program-to-Program Communications (APPC)*.

APPC/MVS

In the z/OS operating system, a session environment that supports LU 6.2 transaction scheduling and communications. The z/OS implementation of APPC.

application programming interface (API)

A formally defined programming language interface between an IBM system control program or a licensed program and the user of a program.

architecture

A set of defined terms and rules used as instructions to build products.

ascending key sequence

Specifies that the records of a file are in ascending key sequence. If the key class is BYTSTRDR (byte string), the collating sequence is a simple binary sequence with X'00' as the lowest value and X'FF' as the highest value.

ASCH

The APPC/MVS scheduler.

ASCII

American National Standard Code for Information Interchange

ASID

Address space identifier

associated DDM attributes

Associated DDM attributes are defined in DDM. Examples of associated DDM attributes are file size, lock options or end-of-file offset for byte-stream files. Associated DDM attributes are not necessarily exclusive to DDM, but can be common to other applications that access the same data sets.

automatic class selection (ACS)

A mechanism for assigning Storage Management Subsystem classes and storage groups to data sets.

automatic class selection (ACS) routine

A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for a data set.

backup

The process of creating a copy of a data set or object to be used in case of accidental loss.

base data set

Data set or file stored on z/OS, in contrast to the view of the file as seen by the workstation. Is also used to refer to the VSAM ESDS or KSDS upon which an alternate index is built.

BCP

Base control program

BDAM

Basic direct access method

BSAM

Basic sequential access method

byte

The amount of storage required to represent one character; the basic unit of data.

byte stream

A simple sequence of bytes stored in a stream file.

C language

A language used to develop software applications in compact, efficient code that can be run on different types of computers with minimal change.

CCSID

Coded character set identifier

CD

Change directory

CDRA

Character Data Representation Architecture

CL

Control language

client

(1) A user. (2) A consumer of resources or services. (3) A functional unit that receives shared services from a server. (4) A system that is dependent on a server to provide it with programs or access to programs. (5) On a network, the computer requesting services or data from another computer.

client-server

(1) In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and waits for a response. The requesting program is called a client; the answering program is called a server. (2) A model of computer interaction in which a server provides resources for other systems on a network, and a client accesses those resources. See also *client*, *server*.

code point

Specifies the data representation of a dictionary code point. Code points are hexadecimal aliases for the named terms of DDM architecture. Code points are used to reduce the number of bytes required to identify the class of an object in memory and in data streams.

command

A message sent to an object requesting that the object carry out one of its operations.

communications manager

Manages the use of the system's communication facilities.

conversation

In Advanced Program-to-Program Communications (APPC), a connection between two transaction programs over a logical unit-logical unit (LU-LU) session that allows them to communicate with each other while processing a transaction.

conversational transaction

In Advanced Program-to-Program Communications (APPC), two or more programs communicating using the services of logical units (LUs).

cursor

A cursor is a displayed symbol that acts as a marker to help the user locate a point in text, in a system command, or in storage. Cursors mark file position and access information in Distributed Data Management architecture.

DASD volume

A DASD space identified by a common label and accessed by a set of related addresses.

data class

A collection of allocation and space attributes, defined by the storage administrator, that are used to create a data set.

data management services

The storage, organization, and access of data.

data set

In DFSMS, the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. In z/OS non-UNIX environments, the terms *data set* and *file* are generally equivalent and sometimes are used interchangeably. See also *file*. In z/OS UNIX environments, the terms *data set* and *file* have quite distinct meanings.

data stream

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

DCAS

DFM central address space

DDM

See *Distributed Data Management Architecture*.

DDM file name

Distributed Data Management file name.

device name

This term is used interchangeably with device number, unit number, and unit name. It is the number by which a specific device is known. For example, an installation with two tape drives might assign them device names 181 and 182.

DFM

Distributed FileManager

DFSMS

See *Data Facility Storage Management Subsystem*.

DFSMSdfp

A DFSMS functional component or base element of z/OS, that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss

A DFSMS functional component or base element of z/OS, used to copy, move, dump, and restore data sets and volumes.

DFSMShsm

A DFSMS functional component or base element of z/OS, used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

DFSMSrmm

A DFSMS functional component or base element of z/OS, that manages removable media.

direct file

A file that contains records that have a relationship between the contents of the record and the record position at which the record is stored.

directory

A file that maps the names of other directories and files to their locations.

distributed computing

Computing that involves the cooperation of two or more machines communicating over a network. Data and resources are shared among the individual computers.

distributed data

Data that is stored in more than one system in a network and is available to remote users and application programs.

distributed data management

A methodology that allows data on one system to be shared and accessed by another system.

Distributed Data Management Architecture (DDM)

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

distributed file

A file that can be accessed by remote applications or remote users. Also, the capability of accessing such a file.

Distributed FileManager

Distributed FileManager (or DFM) is an implementation of target (server) support as defined by Distributed Data Management Architecture (DDM). DDM permits systems in an extended enterprise that have DDM source (client) capability to access file data on a DDM target z/OS system. See definitions for *source*, *target*, and *extended enterprise*.

distributed processing

A capability that enables applications and data located at remote sites or processors connected by a communications link to be used as if they were local.

DSAS

Data space address space

DSS

Data set services

EBCDIC

Extended binary coded decimal interchange code

extended enterprise

A heterogeneous computing environment that often includes both centralized hosts and distributed workstations connected in a network. Gateways within the extended enterprise provide connections to local area networks (LANs). These LANs can serve any computing systems architecture.

ESDS

Entry-sequenced data set

extent

A file extent is a storage area for records allocated to a file by the server. Extents are not formally architected in DDM.

file

A collection of information treated as a unit. In z/OS non-UNIX environments, the terms *data set* and *file* are generally equivalent and are sometimes used interchangeably. See also *data set*.

file class

Refers to the DDM file class (FILCLS) used when writing VSAM for a DDM client or VSAM for AIX applications.

file model

A description of how information is organized and managed within a file.

fixed-length record

A fixed-length record is one whose length is established as an attribute of the file in which it is stored, and can not be changed. Every record in such a file has the same length, which is specified by the record length attribute (RECLLEN) of the file.

gateway

A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures.

GDG

Generation data group

GDS

Generation data set

heterogeneous computer network

A computer network in which computers have dissimilar architecture, but nevertheless are able to communicate.

HFS

Hierarchical file system

ICF

See *Integrated catalog facility (ICF)*.

IDCAMS

Integrated catalog access method services

integrated catalog facility (ICF)

In the Data Facility Product (DFP), a facility that provides for integrated catalog facility catalogs.

Interactive Storage Management Facility (ISMF)

The interactive interface of DFSMS that allows users and storage administrators access to the storage management functions.

IPL

Initial program load

ISMF

See *Interactive Storage Management Facility (ISMF)*.

JCL

job control language

keyed field

The portion of a record which is used (possibly with other key fields) to locate a data record in a keyed file.

KSDS

Key-sequenced data set

LAN

See *local area network*.

LDMI

Local data management interface.

LDS

Linear data set (VSAM)

LE

Language environment

local

Local is your reference point when discussing such entities as platforms or applications. For example, when discussing network conversations from the reference point of an z/OS platform, local refers to entities located on the z/OS system. Similarly, when discussing data access methods from the reference point of an z/OS platform, local refers to the z/OS access methods. Contrast with *remote*.

local area network (LAN)

A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation.

local location name

The name by which a system is known to other systems in an SNA network. A local location name is equivalent to an SNA local logical unit name.

locking

The process of restricting resources to provide protection from concurrent users of the system.

logical unit (LU)

In SNA, a logical port through which an end user accesses the SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services.

logical unit 6.2 (LU 6.2)

A particular type of Systems Network Architecture (SNA) logical unit (LU) that provides a connection between resources and transactions programs running on different network nodes.

LU

See *logical unit*.

LU 6.2

See *logical unit 6.2*.

mainframe

A large computer, particularly one to which other computers can be connected so that they can share facilities the mainframe provides.

management class

A named collection of management attributes describing the retention, backup, and class transition characteristics for a group of objects in an object storage hierarchy.

migration

The process of moving unused data to lower cost storage in order to make space for high-availability data. If you wish to use the data set, it must be recalled. See also *migration level 1* and migration level 2.

migration level 1

DFSMSHsm-owned DASD volumes that contain data sets migrated from primary storage volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage* and *migration level 2*.

migration level 2

DFSMSHsm-owned tape or DASD volumes that contain data sets migrated from primary storage volumes or from migration level 1 volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage* and *migration level 1*.

mode name

The name used by the initiator of a session to designate the characteristics desired for the session, such as traffic pacing values, message-length limits, sync point and cryptography options, and the class of service within the transport network.

object storage hierarchy

A hierarchy consisting of objects stored in DB2® table spaces on DASD, on optical or tape volumes that reside in a library, and on optical or tape volumes that reside on a shelf. See also *storage hierarchy*.

optical volume

Storage space on an optical disk, identified by a volume label. See also *volume*.

partitioned data set (PDS)

A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE)

A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

partner

In data communications, the remote application program or the remote computer. Also refers to complementary information or function on a remote platform. To conduct a network conversation, for example, Distributed FileManager requires a local logical unit (LU) on the target system and a partner LU on the source system.

PDS

See *Partitioned data set*.

PDSE

See *Partitioned data set extended*.

platform

A computer system running a specific operating system connected in a network. For example, z/OS and i5/OS are different operating system platforms.

PPT

Program property table

primary space allocation

Amount of space requested by a user for a data set when it is created. Contrast with *secondary space allocation*.

primary storage

A DASD volume available to users for data allocation. The volumes in primary storage are called primary volumes. See also *storage hierarchy*. Contrast with *migration level 1* and *migration level 2*.

protocol

(1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) A specification for the format and relative timing of information exchanged between communicating parties.

QSAM

Queued sequential access method

RACF

See *Resource Access Control Facility (RACF)*.

record

The basic unit of data stored in a record-oriented file.

record data

Data sets with a record-oriented structure, which are accessed record by record. This data set structure is typical of data sets on VM, z/OS, and i5/OS systems.

record-level access

A means of supporting distributed files. Enables an application or user to read and update individual records of files on a remote system without specifying the data's location.

record-oriented file

File with a record-oriented structure that is accessed record by record. This file structure is typical of data sets on VM, z/OS, and i5/OS systems. Contrast with *stream-oriented file*.

remote

Remote is relative to your reference point when discussing such entities as platforms or applications. For example, when discussing network conversations from the reference point of an z/OS platform, remote refers to entities that access z/OS data across an network. A DDM client application accessing the z/OS data would be remote. Contrast with *local*.

Resource Access Control Facility (RACF)

An IBM licensed program that is included in z/OS Security Server and is also available as a separate program for the z/OS and VM environments. RACF provides access control by identifying and verifying the users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

RLS

Record-level sharing

RRDS

Relative record data set

SAM

Sequential access method

SDDM/400

DDM source on i5/OS

SdU

See *SMARTdata UTILITIES*.

secondary space allocation

Amount of additional space requested by the user for a data set when primary space is full. Contrast with *primary space allocation*.

sequential file

A type of z/OS file that has its records stored and retrieved according to their physical order within the file. It must be on a direct access volume.

server

(1) A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server. (2) On a network, the computer that contains the data or provides the facilities to be accessed by other computers in the network. (3) A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system.

session

A logical connection between two stations or network addressable units (NAUs) that allows them to communicate.

SMARTdata Utilities (SdU)

SMARTdata Utilities (SdU) is a component of a DDM client system that provides source DDM services.

SMS

See *Storage Management Subsystem (SMS)*.

SNA

See *Systems Network Architecture*.

source

Source is the term used in Distributed Data Management Architecture (DDM) to refer to the platform that originates a request for remote data. Source is also known as client. Contrast with *target*.

source server

DDM term for the function that converts source requests to data streams containing DDM commands and output data and sends them over the network to the target server.

source system

A system containing an application program that requests access to data in another system.

SPE

Small programming enhancement

storage administration group

A centralized group within the data processing center that is responsible for managing the storage resources within an installation.

storage administrator

A person in the data processing center who is responsible for defining, implementing, and maintaining storage management policies.

storage class

A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage hierarchy

An arrangement of storage devices with different speeds and capacities. The levels of the storage hierarchy include main storage (memory, DASD cache), primary storage (DASD containing uncompressed data), migration level 1 (DASD containing data in a space-saving format), and migration level 2 (tape cartridges containing data in a space-saving format). See also *primary storage*, *migration level 1*, *migration level 2*, and *object storage hierarchy*.

storage management

The activities of data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These can be done either manually or by using automated processes. The Storage Management Subsystem automates these processes for you, while optimizing storage resources. See also *Storage Management Subsystem*.

Storage Management Subsystem (SMS)

A DFSMS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

Stream data file

Data sets with a byte-oriented structure, which are accessed as continuous streams of data bytes. This data set (file) structure is common in workstation environments.

stream-oriented file

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

system administrator

The person at a computer installation who designs, controls, and manages the use of the computer system.

system operator

An operator responsible for performing system-oriented procedures.

system programmer

A programmer who plans, generates, maintains, extends, and controls the use of an operating system and applications with the aim of improving overall productivity of an installation.

system-managed storage

Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, and space to applications. See also *system-managed storage environment*.

system-managed storage environment

An environment that helps automate and centralize the management of storage. This is achieved through a combination of hardware, software, and policies. In the system-managed storage environment for z/OS, the function is provided by DFSORT, RACF, and the combination of DFSMS and z/OS.

Systems Network Architecture (SNA)

The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the networks and also operational sequences for controlling the configuration and operation of networks.

tape volume

A tape volume is the recording space on a single tape cartridge or reel. See also *volume*.

target

Target is the term used in Distributed Data Management Architecture (DDM) to refer to the platform that fulfills a request for remote data. Target is also known as server. Contrast with *source*.

target server

DDM term that describes the function that converts DDM data streams received from a source server to local data management requests and sends reply messages and input data back to the source server over a network.

target system

A system containing data that has been requested by another system.

TCP/IP

See *Transmission Control Protocol/Internet Protocol (TCP/IP)*.

TP

See *Transaction program (TP)*.

transaction program (TP)

A program that uses the Advanced Program-to-Program Communications (APPC) application programming interface (API) to communicate with a partner application program on a remote system.

Transmission Control Protocol/Internet Protocol (TCP/IP)

The two fundamental protocols of the Internet protocol suite. The abbreviation TCP/IP is frequently used to refer to this protocol suite. TCP/IP provides for the reliable transfer of data, while IP transmits the data through the network in the form of datagrams. Users can send mail, transfer files across the network, or execute commands on other systems.

TSO

Time Sharing Option

user interface

(1) The means by which a user communicates with a system, program, or device. (2) The hardware, software, or both that implements a user interface, allowing the user to interact with and perform operations on a system, program, or device. Examples are a keyboard, mouse, command language, or windowing subsystem.

variable-length record

A variable-length record is one whose length can be changed after it has been written to a file. The length of individual records in the file varies from record to record but cannot exceed the maximum length specified by the RECLEN attribute of the file. The length of a record is initially set by the DDMInsertRecEOF, DDMInsertRecNum or DDMInsertRecKey function, but can be changed by a subsequent function (DDMModifyRec, DDMInsertRecNum, DDMInsertRecKey, or DDMDeleteRec).

volume

The storage space on DASD, tape, or optical devices, which is identified by a volume label. See also *DASD volume*, *optical volume*, and *tape volume*.

VRRDS

Variable-length relative record data set

VSAM

Virtual Storage Access Method

VSE

Virtual Storage Extended

VTAM

Virtual Telecommunications Access Method

wild card

A character or sequence of characters that can be included in a character string to represent zero or more characters in the string.

workstation

(1) A device that enables users to transmit information to or receive information from a computer; for example, a display station or printer. (2) A functional unit at which a user works. It can be a programmable workstation, such as an IBM xSeries computer, or a nonprogrammable workstation, such as a terminal. (3) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

WTO

Write-to-Operator

Index

A

- ACBNAME parameter
 - APPL definition [35](#)
 - LUADD definition [32](#)
- accessibility
 - contact IBM [123](#)
 - features [123](#)
- accessing
 - data in extended enterprise [2](#)
 - z/OS data sets requirements [11](#)
- ACS (automatic class selection)
 - routines for Distributed FileManager
 - data class [41](#)
 - management class [41](#)
 - storage class [41](#)
- activating Distributed FileManager, example [41](#)
- adding to VSAM KSDS
 - side information [34](#)
 - TP profile [33](#)
- administration utility
 - adding TP profile [33](#)
- administrator, TP
 - defining to RACF [43](#)
- Advanced Program-to-Program Communication (APPC) [32](#)
- AGENT parameter [26](#)
- alias names
 - directory access [17](#)
 - record access [15](#)
 - stream access [16](#)
- allocating VSAM KSDS
 - for side information [34](#)
 - for TP profile [33](#)
- altering CCSID parameter [21](#)
- alternate index files
 - base data sets [18](#)
 - DDM file model [12](#)
 - defining [18](#)
- APPC (Advanced Program-to-Program Communication)
 - Distributed FileManager
 - creating side information [34](#)
 - creating TP profile [33](#)
 - defining APPC/MVS start parameters [32](#)
 - defining APPC/MVS transaction scheduler [34](#)
 - LUADD definition [32](#)
 - TP profile [32](#)
 - starting
 - APPC/MVS [45](#)
 - APPC/MVS transaction scheduler [45](#)
 - stopping APPC/MVS [48](#)
 - support for Distributed FileManager [6](#)
 - using administration utility [33](#)
- APPL definition [35](#)
- ASCH
 - defining start parameters [34](#)
 - displaying status [46](#)

- ASCH (*continued*)
 - stopping [48](#)
- assistive technologies [123](#)
- ATBSDFMU [33](#)
- attributes, file, DDM [113](#)
- automatic class selection (ACS) [41](#)

B

- BASE parameter, LUADD definition [32](#)

C

- CANCEL command
 - controlling conversations [48](#)
- CCSID (coded character set identifier)
 - altering [22](#)
 - determining
 - using IDCAMS [24](#)
 - using ISMF [23](#)
 - introduction [21](#)
- CDRA (character data representation architecture)
 - introduction [21](#)
- Character Data Representation Architecture (CDRA) [21](#)
- CLASSADD definition
 - defining start parameters, transaction scheduler [34](#)
- client/server relationship [4](#)
- coded character set identifier [22](#)
- contact
 - z/OS [123](#)
- converting data [22](#)
- creating DDM file attributes [23](#)
- customizing
 - for Distributed FileManager
 - APPC/MVS start parameters [32](#)
 - APPC/MVS transaction scheduler [34](#)
 - illustration of tasks [29](#)
 - startup parameters [38](#)
 - startup procedure [41](#)
 - summary of tasks [29](#)
 - VTAM [35](#)

D

- data set
 - altering REUSE parameter [19, 20](#)
 - data conversion [22](#)
 - Distributed FileManager tuning parameters [40](#)
 - name mapping [18](#)
 - naming, using Distributed FileManager [18](#)
 - using Distributed FileManager
 - access limitations [15, 16](#)
 - access requirements [11](#)
 - z/OS
 - supported by Distributed FileManager [11](#)
- DataAgent

- DataAgent (*continued*)
 - accessing data [26](#)
 - DFMQTSO sample [72](#)
 - DFMX0001 sample [55](#)
 - DFMXAGNT sample [56](#)
 - DFMXSORT sample [61](#)
 - DFMXSRTI sample [68](#)
 - DFMXTSOI sample [78](#)
 - how it works [9](#)
- dbtoken
 - defining for Distributed FileManager [43](#)
- DDM (Distributed Data Management Architecture)
 - file attributes
 - altering CCSID parameter [21](#)
 - creating [23](#)
 - definition [23](#)
 - determining [23](#), [24](#)
 - loss of [15](#), [16](#)
 - propagating [25](#)
 - implementation [5](#)
 - implementations
 - platforms supporting [6](#)
 - relationship to Distributed FileManager [5](#)
 - source, communicating with Distributed FileManager [8](#)
- DDM client system
 - defining
 - local LU, example [38](#)
- DDM file attributes [113](#)
- DDM file models [12](#)
- defining
 - ACS routines for Distributed FileManager
 - data class example [41](#)
 - management class example [41](#)
 - storage class example [41](#)
 - APPC/MVS start parameters
 - LUADD definition [32](#)
 - Distributed FileManager
 - side information [34](#)
 - TP profile [33](#)
 - logon mode to VTAM [36](#)
 - partner information, requirements [37](#)
 - startup procedure
 - for Distributed FileManager [41](#)
 - to RACF
 - TP administrator [43](#)
 - user ID [44](#)
- determining DDM file attributes
 - using DCOLLECT [24](#)
 - using ISMF [23](#)
 - using LISTCAT [24](#)
- DFM00 system sample [51](#)
- DFMQTSO sample [72](#)
- DFMX0001 sample [55](#)
- DFMXAGNT sample [56](#)
- DFMXSORT sample [61](#)
- DFMXSRTI sample [68](#)
- DFMXTSOI sample [78](#)
- directories
 - access [16](#)
 - restrictions [17](#)
- DISPLAY APPC command
 - example
 - LU status [46](#)
 - TP status [46](#)
- DISPLAY ASCH command [46](#)
- displaying status
 - local LU [46](#)
 - transaction program (TP) [46](#)
 - transaction scheduler [46](#)
- Distributed Data Management Architecture (DDM) [5](#)
- distributed data processing
 - client/server relationship [4](#)
 - discussion [3](#)
- Distributed FileManager
 - access requirements [11](#)
 - accessing data using the DataAgent [26](#)
 - APPC/MVS support, overview [6](#)
 - applications of [10](#)
 - benefits [4](#)
 - communicating with DDM source [8](#)
 - controlling conversations
 - stopping local LU [47](#)
 - stopping TP [47](#), [48](#)
 - creating
 - side information [34](#)
 - TP profile [33](#)
 - customizing
 - APPC/MVS [31](#)
 - illustration of tasks [29](#)
 - summary of tasks [29](#)
 - VTAM [35](#)
 - data conversion [22](#)
 - data set
 - access limitations [15](#), [16](#)
 - name mapping [18](#)
 - naming [18](#)
 - DDM file attributes
 - altering CCSID parameter [21](#)
 - definition [23](#)
 - propagating PDSE [25](#)
 - propagating PS and VSAM [25](#)
 - using IDCAMS to determine [24](#)
 - using ISMF to determine [23](#)
 - DDM source, relationship with [6](#)
 - DDM support, overview [5](#)
 - defining
 - APPC/MVS transaction scheduler [34](#)
 - partner information [37](#)
 - startup parameters for Distributed FileManager for z/OS [38](#)
 - startup procedure [41](#)
 - how DataAgent works [9](#)
 - monitoring conversations [46](#)
 - operating procedures [45](#)
 - processing environment
 - components [5](#)
 - description [7](#)
 - RACF support, overview [6](#)
 - starting [45](#)
 - stopping [48](#)
 - TP access security
 - requirements [43](#)
 - using RACF [43](#)
 - verifying PPT entries [41](#)
 - VTAM support, overview [7](#)
 - wild card options [18](#)
 - z/OS data sets
 - not supported [12](#)

Distributed FileManager (*continued*)
z/OS data sets (*continued*)
supported [11](#)

E

EXPORT command
propagating DDM attributes [25](#)
extended enterprise, data access in [2](#)

F

feedback [xv](#)
file attributes, DDM [113](#)
FORCE command, stopping TP [48](#)

G

gateway, in extended enterprise [2](#)
GDEAPDEF system sample [49](#)
GDEAPPC system sample [49](#)
GDEASCH system sample [50](#)
GDELOGMD system sample [51](#)
GDEPRTLUI system sample [54](#)
GDETPDEF system sample [53](#)

H

hidden files [17](#)

I

IDCAMS
altering REUSE parameter [19](#)
determining DDM attributes
using DCOLLECT command [24](#)
using LISTCAT command [24](#)
propagating DDM attributes
using IMPORT command [25](#)
IEBCOPY utility
propagating DDM attributes
for PDSE member [25](#)
implementing
TP security to RACF [44](#)
IMPORT command
propagating DDM attributes [25](#)
Interactive Storage Management Facility (ISMF) [23](#)
ISMF (Interactive Storage Management Facility)
determining DDM file attributes [23](#)

K

keyboard
navigation [123](#)
PF keys [123](#)
shortcut keys [123](#)

L

LISTCAT command
determining DDM attributes [24](#)
local LU [35](#)

logical unit (LU)
local
APPL definition [35](#)
DDM client, example [38](#)
defining to APPC/MVS [32](#)
defining to VTAM [35](#)
displaying status [46](#)
stopping [47](#)
partner
defining to VTAM [37](#)
logon mode
defining to VTAMLIB [36](#)
defining to VTAMLST [35](#)
logon mode requirements [121](#)
LU [35](#)
LU 6.2 protocol
description [6](#)
LUADD definition
example [32](#)

M

mapping data set name on a DDM client [18](#)
migrated files [17](#)
multivolume data sets [15](#)

N

naming data set
using Distributed FileManager [18](#)
navigation
keyboard [123](#)

P

parameters, DataAgent file name suffix [26](#)
PARM parameter [26](#)
PARMLIB
APPCPMxx member
defining APPC/MVS start parameters [32](#)
LUADD definition [32](#)
ASCHPMxx member
defining transaction scheduler [34](#)
DFM00 member
Distributed FileManager startup parameters [38](#)
SCHEDxx member [41](#)
partner LU [35](#)
PDSE member
propagating DDM attributes
restrictions [25](#)
PDSEs
advantages [21](#)
using [21](#)
PDSs
limitations [21](#)
using [21](#)
performance
Distributed FileManager
tuning parameters [39](#)
PGM parameter [27](#)
platform, operating system
DDM implementations on [6](#)
PPT (program property table)

- PPT (program property table) (*continued*)
 - verifying Distributed FileManager entries [41](#)
- PPT Entries for DFM for z/OS [85](#)
- preserving DDM attributes [25](#)
- PROCLIB
 - DFM member
 - Distributed FileManager startup procedure [41](#)
- program property table (PPT) [41](#)
- propagating DDM attributes
 - for PDSE [25](#)
 - for PS and VSAM data sets [25](#)

Q

- QPCSUPP logon mode [121](#)

R

- RACF (Resource Access Control Facility)
 - defining TP access security
 - description [6](#)
 - examples [43](#)
 - for Distributed FileManager [43](#)
- record-oriented data
 - access restrictions using Distributed FileManager [15](#)
 - DDM access functions [14](#)
 - DDM access methods [13](#)
 - DDM file classes [13](#)
- reusing VSAM data set [19](#)

S

- SCHED parameter, LUADD definition [33](#)
- sending to IBM
 - reader comments [xv](#)
- SET command
 - controlling TP status [47](#)
- shortcut keys [123](#)
- SIDEINFO DATASET parameter
 - in LUADD definition [32](#)
- SNA (Systems Network Architecture)
 - LU 6.2 protocol [6](#)
- source
 - relationship with target [4](#)
- START parameter [27](#)
- starting
 - APPC/MVS [45](#)
- starting up
 - APPC/MVS transaction scheduler [45](#)
 - Distributed FileManager [45](#)
- stream-oriented data
 - access restrictions
 - using Distributed FileManager [16](#)
 - DDM access functions [15](#)
 - DDM access method [16](#)
 - DDM file class [15](#)
- summary of changes [xvii](#)
- system files [17](#)
- system PROCLIB member DFM, sample [83](#)
- system samples
 - DFM00 [51](#)
 - DFMQTSO [72](#)
 - DFMX0001 [55](#)

- system samples (*continued*)

- DFMXAGNT [56](#)
- DFMXSORT [61](#)
- DFMXSRTI [68](#)
- DFMXTSOI [78](#)
- GDEAPDEF [49](#)
- GDEAPPC [49](#)
- GDEASCH [50](#)
- GDELOGMD [51](#)
- GDEPRTL U [54](#)
- GDETPDEF [53](#)

- Systems Network Architecture (SNA) [6](#)

T

- target
 - relationship with source [4](#)
- terminology, special [1](#)
- TP (transaction program) [32](#)
- TPDATA parameter
 - in LUADD definition [32](#)
- TPMODIFY command
 - changing TP status [47](#)
- transaction program (TP)
 - access security
 - defining to RACF [44](#)
 - requirements for [43](#)
 - defining for Distributed FileManager [33](#)
 - description [32](#)
 - displaying status [46](#)
 - profile
 - defining to RACF [43](#)
 - stopping [47](#), [48](#)
- transaction scheduler, APPC/MVS
 - defining start parameters [34](#)
 - starting [45](#)
- tuning parameters
 - for Distributed FileManager
 - related to data set definition [40](#)
 - related to performance [39](#)

U

- user ID, defining to RACF [44](#)
- user interface
 - ISPF [123](#)
 - TSO/E [123](#)

V

- verifying PPT entries
 - for Distributed FileManager [41](#)
- Virtual Telecommunications Access Method (VTAM) [35](#)
- VSAM data set
 - altering REUSE parameter [19](#)
 - propagating DDM attributes [25](#)
- VTAM (Virtual Telecommunications Access Method)
 - APPL definition [35](#)
 - defining
 - local LU to VTAMLST [35](#)
 - logon mode to VTAMLIB [36](#)
 - partner information [37](#)
 - support, overview [7](#)

VTAMLIB
 defining logon mode [36](#)
VTAMLST
 defining local LU [35](#)

W

wild cards
 restrictions [18](#)
workstation
 DFSMSdfp support [4](#)



Product Number: 5650-ZOS

SC23-6848-50

