z/OS
2.5

DFSMSrmm Application Programming Interface

SC23-6872-50

Note

Before using this information and the product it supports, read the information in “Notices” on page 159.

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: 2023-05-15

Contents

Figures

Tables

About this document

This document is intended for application programmers who use the DFSMSrmm application programming interface to obtain information about resources that are managed by DFSMSrmm.

Refer to:

- Chapter 1, "Using the DFSMSrmm application programming interface," on page 1 for information on the EDGXCI macro you use for communication between your application program and DFSMSrmm.
- Chapter 2, "Using the object-oriented DFSMSrmm application programming interface using C++," on page 15 for information on using C++ and other high-level programming languages to write programs to obtain information about DFSMSrmm resources.
- Chapter 3, "Using the DFSMSrmm application programming interface using assembler language," on page 23 for guidelines for using the application programming interface.
- Chapter 5, "Processing the output data in the output buffer," on page 35 for information on the data that the DFSMSrmm application programming interface returns.

For information about accessibility features of z/OS, for users who have a physical disability, see Appendix E, "Accessibility," on page 157.

Required product knowledge

To use this document effectively, you should be familiar with:

- The RMM TSO subcommand and operands
- Macros to communicate between programs

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).

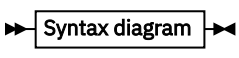
Notational conventions

This section explains the notational conventions used in this document.

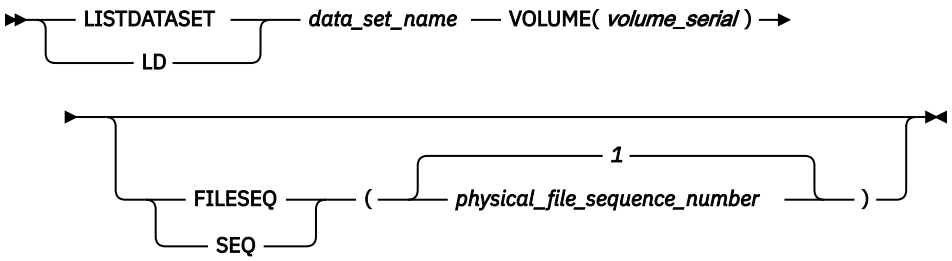
How to read syntax diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. Keyword parameters are parameters that follow the positional parameters. Unless otherwise stated, keyword parameters can be coded in any order. The following list tells you how to interpret the syntax diagrams:

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with two arrowheads facing each other.

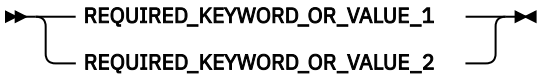


- If a diagram is longer than one line, each line to be continued ends with a single arrowhead and the next line begins with a single arrowhead.



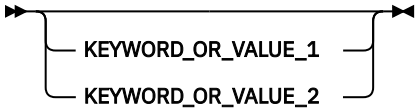
- Required keywords and values appear on the main path line. You must code required keywords and values.

  If several mutually exclusive required keywords or values exist, they are stacked vertically in alphanumeric order.



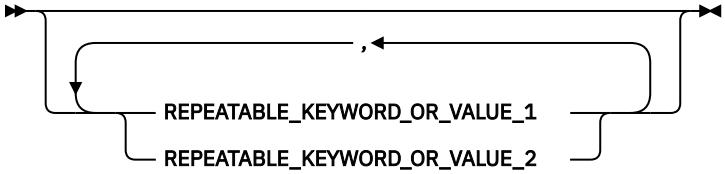
- Optional keywords and values appear below the main path line. You can choose not to code optional keywords and values.

  If several mutually exclusive optional keywords or values exist, they are stacked vertically in alphanumeric order below the main path line.



- An arrow returning to the left above a keyword or value on the main path line means that the keyword or value can be repeated. The comma means that each keyword or value must be separated from the next by a comma.

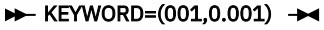
- An arrow returning to the left above a group of keywords or values means more than one can be selected, or a single one can be repeated.



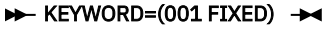
- A word in all uppercase is a keyword or value you must spell exactly as shown. In this example, you must code ***KEYWORD***.

  If a keyword or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.

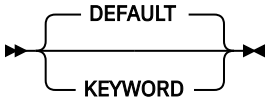
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***KEYWORD=(001,0.001)***.

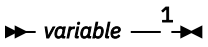
- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***KEYWORD=(001 FIXED)***.

- Default keywords and values appear above the main path line. If you omit the keyword or value entirely, the default is used.

- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

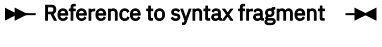
  Notes:

  [1] An example of a syntax note.

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

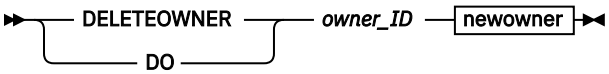
- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

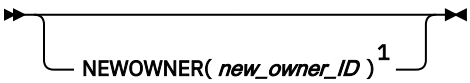
  Syntax fragment

The following is an example of a syntax diagram.



newowner



Notes:

[1] Must be specified if the owner owns one or more volumes.

The possible valid versions of the RMM DELETEOWNER command are:

```
RMM DELETEOWNER owner
RMM DO          owner
RMM DELETEOWNER owner NEWOWNER(new_owner)
RMM DO          owner NEWOWNER(new_owner)
```

How to abbreviate commands and operands

The TSO abbreviation convention applies for all DFSMSrmm commands and operands. The TSO abbreviation convention requires you to specify as much of the command name or operand as is necessary to distinguish it from the other command names or operands.

Some DFSMSrmm keyword operands allow unique abbreviations. All unique abbreviations are shown in the command syntax diagrams.

How to use continuation characters

The symbol **-** is used as the continuation character in this document. You can use either **-** or **+**.

-
Do not ignore leading blanks on the continuation statement

+
Ignore leading blanks on the continuation statement

Delimiters

When you type a command, you must separate the command name from the first operand by one or more blanks. You must separate operands by one or more blanks or a comma. Do not use a semicolon as a delimiter because any character you enter after a semicolon is ignored.

Character sets

To code job control statements, use characters from the character sets in Table 1 on page xiv. Table 2 on page xiv lists the special characters that have syntactical functions in job control statements.

Table 1. Character sets

Character Set	**Contents**	
Alphanumeric	Alphabetic Numeric	Capital A through Z 0 through 9
National (See note)	“At” sign Dollar sign Pound sign	@ (Characters that can be $ represented by hexadecimal # values X'7C', X'5B', and X'7B')
Special	Comma Period Slash Apostrophe Left parenthesis Right parenthesis Asterisk Ampersand Plus sign Hyphen Equal sign Blank	, . / ' ( ) * & + - =
EBCDIC text	EBCDIC printable character set	Characters that can be represented by hexadecimal X'40' through X'FE'
Note: The system recognizes the following hexadecimal representations of the U.S. National characters; @ as X'7C'; $ as X'5B'; and # as X'7B'. In countries other than the U.S., the U.S. National characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the $ character may generate a X'4A'.		

Table 2. Special characters used in syntax

Character	**Syntactical Function**
,	To separate parameters and subparameters

Table 2. Special characters used in syntax (continued)	
Character	**Syntactical Function**
=	To separate a keyword from its value, for example, BURST=YES
(b)	To enclose subparameter list or the member name of a PDS or PDSE
&	To identify a symbolic parameter, for example, &LIB
&&	To identify a temporary data set name, for example, &&TEMPDS, and, to identify an in-stream or sysout data set name, for example, &&PAYOUT
.	To separate parts of a qualified data set name, for example, A.B.C., or parts of certain parameters or subparameters, for example, nodename.userid
*	To refer to an earlier statement, for example, OUTPUT=*.name, or, in certain statements, to indicate special functions: //label CNTL * //ddname DD * RESTART=* on the JOB statement
'	To enclose specified parameter values which contain special characters
(blank)	To delimit fields

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 xvii.

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 (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 DFSMSrmm Application Programming Interface, SC23-6872-50
- 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 (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).

Summary of changes for z/OS Version 2 Release 5

The following changes are made for z/OS Version 2 Release 5 (V2R5). The most recent updates are listed at the top of each section.

Changed

The following content is changed.

- Updated "EDGXSF mapping" on page 92, Appendix B, "Structured field introducers by subcommand," on page 87, "Structured field introducers for subcommand output data" on page 67 and "LISTCONTROL structured field introducers" on page 47.
- DFSMSrmm no longer supports CIM and any information is removed.
- DFSMSrmm no longer supports web services and any information is removed.

Summary of changes for z/OS Version 2 Release 4

The following changes are made for z/OS Version 2 Release 4 (V2R4). The most recent updates are listed at the top of each section.

New

Prior to June 2020 refresh

The following content is new.

- Added new LISTCONTROL structured field introducers DEFZ, DEFN, DEFM, DEFP, DSXP, and CDSF. For more information, see "LISTCONTROL structured field introducers" on page 47.
- Added new LISTDATASET structured field introducer FRXP. For more information, see "LISTDATASET structured field introducers" on page 51.
- Added new LISTVOLUME structured field introducer EXRB and EDM. For more information, see "LISTVOLUME structured field introducers" on page 53.
- Added new command structured field introducers X'1E550', X'1E570', X'1E700', X'1ED00', X'812400', X'82B500', and X'837500'. For more information, see "Structured field introducers for subcommand output data" on page 67.

Changed

Prior to June 2020 refresh

The following content is changed.

- Changed by adding, DEFN, DEFP, and DEFZ to the LISTCONTROL DEFTABLE subcommand, DSXP to the LISTCONTROL OPTION subcommand, CDSF to the LISTCONTROL CNTL subcommand, and FRXP to the

LISTDATASET subcommand. For more information, see Appendix B, “Structured field introducers by subcommand,” on page 87.

- Changed by adding 'ForceExpire' as a possible value of the CTRT SFI. For more information, see “Structured field introducers for subcommand output data” on page 67.

Deleted

No content was removed from this information.

Summary of changes for z/OS Version 2 Release 3

This edition contains updates for Version 2 Release 3 (V2R3).

New information

This edition includes the following new information:

- “Structured field introducers for subcommand output data” on page 67 contains new Catalog Retained SFI information.
- “Structured field introducers for subcommand output data” on page 67 contains new EDM SFI information.

Changed information

This edition includes the following topics that contain changed information:

- “Structured field introducers for subcommand output data” on page 67 was updated with new value PERMANENT.
- Appendix B, “Structured field introducers by subcommand,” on page 87 added SFIs to LISTDATASET and LISTVOLUME subcommands.
- “Begin and End Resource groups” on page 42 added DEFAULT group.
- “LISTCONTROL structured field introducers” on page 47 added SFIs to LISTCONTROL subcommand.
- “Structured field introducers for Begin and End Resource groups” on page 64 added row for DEFAULT - within CONTROL.
- “EDGXSF mapping” on page 92

Chapter 1. Using the DFSMSrmm application programming interface

This topic tells you how to use the application programming interface (API) provided by DFSMSrmm (which is a z/OS feature) to read, extract, and update data in the DFSMSrmm control data set:

- From a high-level language such as C++ or Java™ and receive the output through structured field introducers (SFIs) or XML.
- From assembler language (using EDGXCI) and receive the output by line format or SFI format.

You can use the output data to create reports or implement automation.

For more information about using C++, see:

- Chapter 2, "Using the object-oriented DFSMSrmm application programming interface using C++," on page 15

Use macro EDGXCI as described in "EDGXCI: Calling the DFSMSrmm interface" on page 2 to define a parameter list to call the DFSMSrmm application programming interface. Use macro EDGXCI to pass any supported RMM TSO subcommand to DFSMSrmm. See "Supported RMM TSO subcommands" on page 1 for a list of supported RMM TSO subcommands. "EDGXCI example" on page 12 provides an example that you can modify to communicate with the DFSMSrmm application programming interface.

Use macro EDGXSF as described in "EDGXSF: Structured field definitions" on page 91 to help you process the data that the DFSMSrmm application programming interface returns. The DFSMSrmm application programming interface returns data as structured fields in an output buffer that you define. Structured fields consist of these parts.

- A structured field introducer (SFI) that introduces the type of data, length, and characteristics of the data that the API returns,
- Data.

You can request that the API returns data in line format or field format as described in "Requesting structured field introducer data format" on page 36. You can also request standard output or expanded output as described in "Requesting types of output" on page 39.

To use the DFSMSrmm application programming interface, you must have High Level Assembler installed on your system. *z/OS Planning for Installation* provides information about the level of High Level Assembler required for DFSMS.

Supported RMM TSO subcommands

The DFSMSrmm API supports all the RMM TSO subcommands as shown in Table 3 on page 1.

Table 3. RMM TSO subcommands

Group	Subcommand	Abbrev	Function
Add	ADDBIN	AB	Add bin number information
	ADDDATASET	AD	Add data set information
	ADDOWNER	AO	Add owner information
	ADDPRODUCT	AP	Add software product information
	ADDRACK	AR	Add shelf location information
	ADDVOLUME	AV	Add volume information
	ADDVRS	AS	Add a vital record specification
Change	CHANGEDATASET	CD	Change data set information
	CHANGEOWNER	CO	Change owner information
	CHANGEPRODUCT	CP	Change software product information
	CHANGEVOLUME	CV	Change volume information

Table 3. RMM TSO subcommands (continued)

Group	Subcommand	Abbrev	Function
Delete	DELETEBIN	DB	Delete bin number information
	DELETEDATASET	DD	Delete data set information
	DELETEOWNER	DO	Delete owner information
	DELETEPRODUCT	DP	Delete software product information
	DELETERACK	DR	Delete shelf location information
	DELETEVOLUME	DV	Release a volume and delete volume
	DELETEVRS	DS	Delete a vital record specification information
Get	GETVOLUME	GV	Request or assign a volume
List	LISTBIN	LB	Display bin number information
	LISTCONTROL	LC	Display PARMLIB options and control information
	LISTDATASET	LD	Display data set information
	LISTOWNER	LO	Display owner information
	LISTPRODUCT	LP	Display software product information
	LISTRACK	LR	Display shelf location information
	LISTVOLUME	LV	Display volume information
	LISTVRS	LS	Display vital record specification information
Search	SEARCHBIN	SB	Create a list of bin numbers
	SEARCHDATASET	SD	Create a list of data sets
	SEARCHOWNER	SO	Create a list of owners
	SEARCHPRODUCT	SP	Create a list of software products
	SEARCHRACK	SR	Create a list of rack numbers
	SEARCHVOLUME	SV	Create a list of volumes
	SEARCHVRS	SS	Create a list of vital record specifications

Refer to *z/OS DFSMSrmm Managing and Using Removable Media* for details on these subcommands.

Rule: When you use the DFSMSrmm application programming interface, you must specify the subcommand as a single, continuous string of characters rather than as multiple input lines.

Using the EDGXCI macro

Follow these steps to obtain information from DFSMSrmm using the EDGXCI macro.

1. Use EDGXCI MF=(L,addr) to save space in your dynamic area for the parameter list.
2. Save the address of an output buffer that the application programming interface uses.
3. Load the DFSMSrmm API module, EDGXAPI, and then save the address of the module.
4. Create the subcommand that you want to process.
5. Use the EDGXCI macro to complete the parameter list and call the DFSMSrmm application programming interface.
6. Use EDGXCI with OPERATION=CONTINUE as needed to get more data for the current subcommand.
7. Use EDGXCI with OPERATION=RELEASE to free resources that are obtained by the DFSMSrmm API module.
8. Delete the EDGXAPI module that you loaded.

EDGXCI: Calling the DFSMSrmm interface

Use the EDGXCI macro in your application program (the caller) to:

- Define a parameter list.
- Set parameters in the list.

- Change parameters in the list.
- Call the DFSMSrmm application programming interface module, EDGXAPI.

EDGXCI environment

The requirements for the caller are:

Minimum authorization
Non-APF authorized, problem state and key (0-8).

Dispatchable unit mode
Task

Cross memory mode
PASN=HASN=SASN

AMODE
31-bit

ASC mode
Primary

Interrupt status
Enabled for I/O and external interrupts

Locks
The caller must not be locked.

Control parameters
Control parameters must be in the primary address space.

EDGXCI programming requirements

The caller must load the DFSMSrmm API module, EDGXAPI, prior to using the execute or standard form of EDGXCI. The caller must delete EDGXAPI when the DFSMSrmm API is no longer needed.

The caller should also use the EDGXSF macro to define the structured fields that are used in the output.

See Appendix C, “DFSMSrmm application programming interface mapping macros,” on page 91 for a complete description of the EDGXCI and EDGXSF macros.

EDGXCI restrictions

The caller must not have functional recovery routines (FRRs) established.

The DFSMSrmm API uses Name/Token services to create a non-persistent task-level Name/Token pair for each TOKEN that has not been released. If you plan to use Checkpoint/Restart, refer to the section "Using Checkpoint/Restart with Name/Token Pairs" in *z/OS MVS Programming: Assembler Services Guide*.

EDGXCI input register information

Before issuing the EDGXCI macro, ensure that these general purpose registers (GPRs) contain the specified information:

Register
Contents

13
The address of a 72-byte standard save area in the primary address space

Before issuing the EDGXCI macro, no information is needed in any access register (AR) unless the access register is used in register notation for a particular parameter or as a base register.

EDGXCI output register information

When control returns to the caller, the GPRs contain:

Register
Contents

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When control returns to the caller, the ARs contain:

Register
Contents

0-1
Used as work registers by the system

2-13
Unchanged

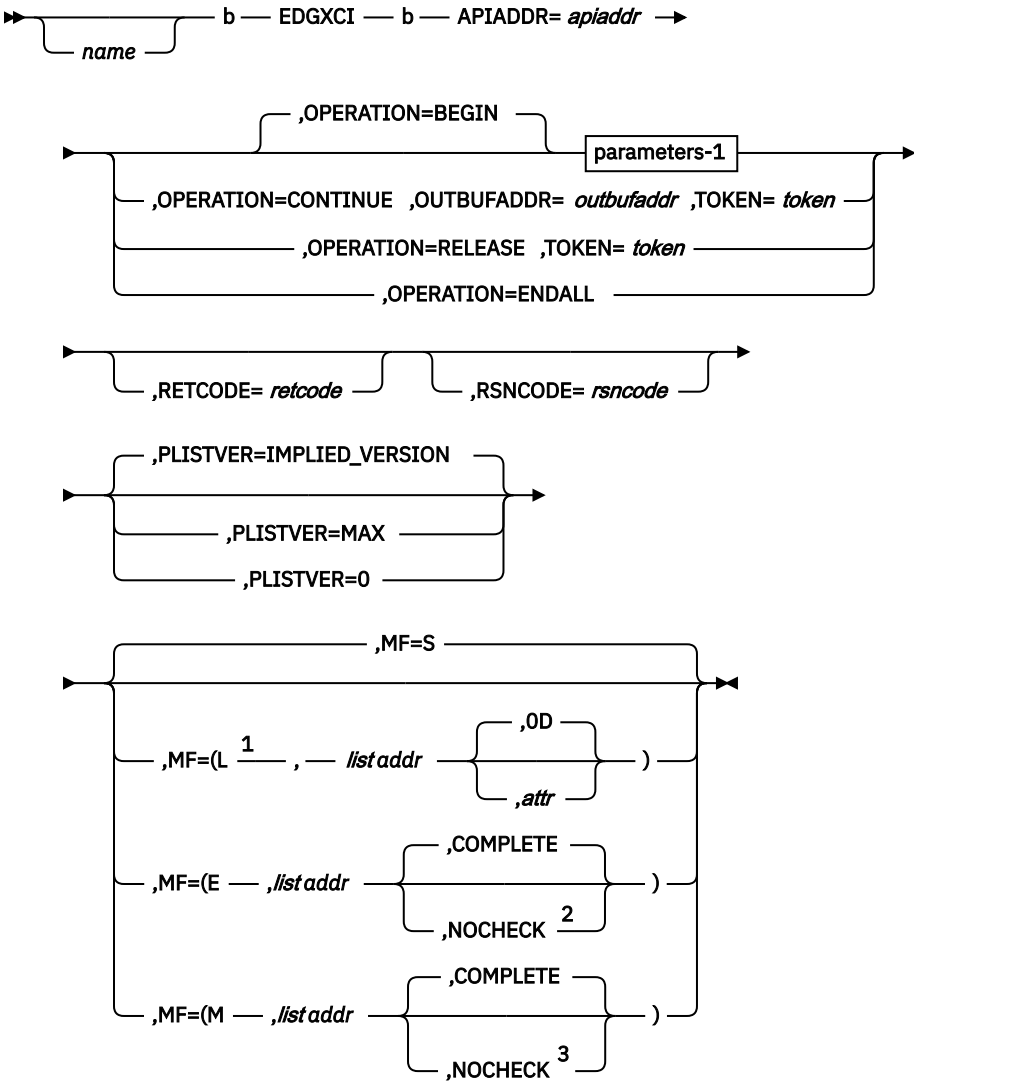
14-15
Used as work registers by the system

Some callers depend on register contents that remain the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

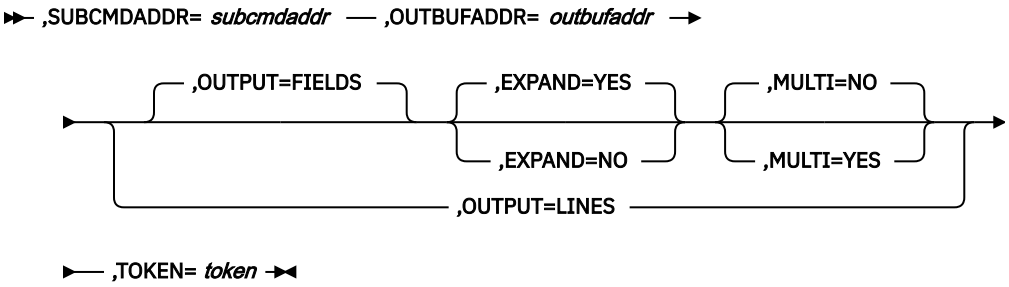
EDGXCI syntax

Figure 1 on page 5 shows the syntax for the EDGXCI macro. You can use this macro to communicate with the DFSMSrmm application programming interface.

EDGXCI macro



parameters-1



Notes:

[1] Only the PLISTVER parameter can be coded with MF=L.

[2] When NOCHECK is specified with MF=E, all parameters are optional and the system does not supply defaults for omitted optional parameters.

[3] When NOCHECK is specified with MF=M, all parameters are optional and the system does not supply defaults for omitted optional parameters.

Figure 1. EDGXCI macro syntax diagram

EDGXCI parameters

You can specify these parameters:

name
An optional symbol that starts in column 1. This is the name on the EDGXCI macro call. The name must conform to the rules for an ordinary assembler language symbol.

APIADDR=*apiaddr*
A required input parameter that contains the address of the DFSMSrmm API load module. The calling program is responsible for loading the DFSMSrmm API load module, saving, and then using the returned load address. Use the z/OS LOAD service to obtain the DFSMSrmm API address.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

EXPAND=NO
EXPAND=YES
When OUTPUT=FIELDS and OPERATION=BEGIN are specified, EXPAND is an optional parameter that specifies whether to expand the number of returned data fields to be the same as for the corresponding list type of subcommand. The default is EXPAND=YES.

EXPAND=NO
Specify to not expand the number of data fields for the subcommand.

EXPAND=YES
Specify to expand the number of data fields to be the same as the corresponding list type of subcommand.

MF=S
MF=(L,*list addr*)
MF=(L,*list addr,attr*)
MF=(L,*list addr*,0D)
MF=(E,*list addr*)
MF=(E,*list addr*,COMPLETE)
MF=(E,*list addr*,NOCHECK)
MF=(M,*list addr*)
MF=(M,*list addr*,COMPLETE)
MF=(M,*list addr*,NOCHECK)
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro. This builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the macro list form with the macro execute form for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form and generates the macro invocation to transfer control to the service.

Use MF=M together with the list form and execute form of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area. Use the modify form to set the appropriate options. Then use the execute form to call the service.

Recommendation: Use the modify and execute forms of EDGXCI in this order:

1. Use EDGXCI ...MF=(M,list-addr,COMPLETE) and specify all the required parameters and any appropriate optional parameters.
2. Use EDGXCI ...MF=(M,list-addr,NOCHECK) and specify the parameters that you want to change.
3. Use EDGXCI ...MF=(E,list-addr,NOCHECK) to execute the macro.

,list addr
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

attr
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of X'0F' to force the parameter list to a word boundary orX'0D' to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of X'0D'.

COMPLETE
Specifies that the system should check for required parameters and supply defaults for omitted optional parameters.

NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

MULTI=NO
MULTI=YES
When OUTPUT=FIELDS and OPERATION=BEGIN are specified, MULTI is an optional parameter that specifies whether a single resource group is to be returned in the buffer, or whether as many resources as fit in the buffer are to be returned. The default is MULTI=NO

MULTI=NO
Specifies that only a single entry can be handled by the API caller.

MULTI=YES
Specifies that multiple entries can be handled by the API caller.

OPERATION=BEGIN
OPERATION=CONTINUE
OPERATION=RELEASE
OPERATION=ENDALL
An optional parameter that describes the processing of the current subcommand. The default is OPERATION=BEGIN.

OPERATION=BEGIN
Specify BEGIN to start a new subcommand.

OPERATION=CONTINUE
Specify CONTINUE to continue the current subcommand.

OPERATION=RELEASE
Specify when you want the token and all its associated resources to be released.

OPERATION=ENDALL
Specify when you want to end all operations by releasing all tokens and all resources.

OUTBUFADDR=*outbufaddr*
When OPERATION=BEGIN is specified, OUTBUFADDR=*outbufaddr* is a required input parameter that contains the address of your output buffer, which is used for both data and messages. It must be at least 4096 bytes in length. The first four bytes of the buffer must contain the length of the buffer, including the four bytes of the length.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

OUTBUFADDR=*outbufaddr*
When OPERATION=CONTINUE is specified, OUTBUFADDR=*outbufaddr* is a required input parameter that contains the address of your output buffer, which is used for both data and messages. It must be at least 4096 bytes in length. The first four bytes of the buffer must contain the length of the buffer, including the four bytes of the length.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

OUTPUT=FIELDS
OUTPUT=LINES
When OPERATION=BEGIN is specified, OUTPUT is an optional parameter that specifies the format of the returned data. The default is OUTPUT=FIELDS.

OUTPUT=FIELDS
Specify when you want data returned in field format.

OUTPUT=LINES
Specify when you want data returned in line format. Search output is always returned in standard form when OUTPUT=LINES is specified.

PLISTVER=IMPLIED_VERSION
PLISTVER=MAX
PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. Specify PLISTVER on all macro forms used for a request and with the same value on all of the macro forms. The PLISTVER values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, which allows you to change to the largest size currently possible. This size might grow from release to release and affect the amount of storage that your application program needs.

  Recommendation: If you can tolerate the size change, always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is large enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of these:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

SUBCMDADDR=*subcmdaddr*
When OPERATION=BEGIN is specified, SUBCMDADDR=*subcmdaddr* is a required input parameter that contains the address of the input subcommand. The subcommand consists of a halfword field followed by the subcommand text. The halfword field must contain the length of the subcommand, including both the halfword field and the subcommand text. The maximum value is 32 761.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

TOKEN=*token*
When OPERATION=BEGIN is specified, TOKEN=*token* is a required input parameter of a 4-byte area. The DFSMSrmm API creates a token and obtains resources for it, or the DFSMSrmm API reuses the token and the resources.

TOKEN is required even when MF=(E,label,NOCHECK) is specified, unless OPERATION=ENDALL is also specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

TOKEN=*token*
When OPERATION=CONTINUE is specified, TOKEN=*token* is a required input parameter of a 4-byte area containing the token used to begin the subcommand. The DFSMSrmm API uses the resources for the token to continue the subcommand.

TOKEN is required even when MF=(E,label,NOCHECK) is specified, unless OPERATION=ENDALL is also specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

TOKEN=*token*
When OPERATION=RELEASE is specified, TOKEN=*token* is a required input parameter of a 4-byte area containing a token. The DFSMSrmm API releases the resources for the token, releases the token, and clears the 4-byte area.

TOKEN is required even when MF=(E,label,NOCHECK) is specified, unless OPERATION=ENDALL is also specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

EDGXCI return and reason codes

When the EDGXCI macro returns control to your application program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- GPR 0 (and *rsncode*, when you code RSNCODE) contains a reason code.

The EDGXCI macro returns these types of return codes and reason codes:

- Return and reason codes that are associated with the processing of your subcommand. These return and reason codes are the same ones that DFSMSrmm returns when you issue a subcommand request. Refer to *z/OS DFSMSrmm Managing and Using Removable Media* for more information about these return and reason codes.
- Return codes and reason codes that are issued by the API. The API returns:
  - Return code 0 and reason code 0 when processing has completed successfully.
  - Return code 0 and reason code 4 when the output buffer is full and more information is available.
  - Any return code higher than 100 when an error has occurred.
- When you use the API with high-level programming languages, DFSMSrmm returns a return code and reason code and a message described in the related messages column in Table 4 on page 9. When you use the standard API, DFSMSrmm does not return a message but you can look to the related message for guidance.

Table 4 on page 9identifies the decimal return and reason codes.

Table 4. Return and reason codes for the EDGXCI macro

Return Code	Reason Code	Meaning and Action	Related Message
0	—	**Meaning**: Success. **Action**: Refer to the action provided with the specific reason code.	
0	0	**Meaning**: EDGXCI command is successfully completed. **Action**: None required.	

Table 4. Return and reason codes for the EDGXCI macro (continued)

Return Code	Reason Code	Meaning and Action	Related Message
0	4	**Meaning**: There is more output waiting to be given to you. **Action**: After you have processed the output in your output buffer, use OPERATION=CONTINUE to get more output.	EDG3900I
104	—	**Meaning**: Program error. An exception condition has been encountered, but the operation you requested was completed. The output results might not be acceptable to you. **Action**: Refer to the action provided with the specific reason code.	
104	02	**Meaning**: There is nothing to CONTINUE. **Action**: None required.	EDG3901I
108	—	**Meaning**: Program error. An error condition has been encountered, and the operation you requested was not successfully completed. **Action**: Refer to the action provided with the specific reason code.	
108	02	**Meaning**: Required token is missing. **Action**: You need to use TOKEN=**token**	EDG3902E
108	04	**Meaning**: Required address of the input subcommand is missing. **Action**: You need to use SUBCMDADDR=**subcmdaddr**	EDG3903E
108	06	**Meaning**: Required address of your output buffer is missing. **Action**: Use OUTBUFADDR=**outbufaddr** to specify the parameter.	EDG3904E
108	08	**Meaning**: Your output buffer is less than 4096 bytes in size. **Action**: Obtain storage and set its length.	EDG3905E
108	10	**Meaning**: Your output buffer is too small. The second word in your buffer contains the size you need. **Action**: Obtain the correct amount of storage and set its length.	EDG3906E
108	12	**Meaning**: OPERATION parameter is invalid. **Action**: Use OPERATION= to specify the parameter; check your program for incorrect modifying of the parameter list.	EDG3907E

Table 4. Return and reason codes for the EDGXCI macro (continued)

Return Code	Reason Code	Meaning and Action	Related Message
108	14	**Meaning**: OUTPUT parameter is invalid. **Action**: Use OUTPUT= to specify the parameter; check your program for incorrect modifying of the parameter list.	EDG3908E
108	16	**Meaning**: EXPAND parameter is invalid. **Action**: Use EXPAND= to specify the parameter; check your program for incorrect modifying of the parameter list.	EDG3909E
108	18	**Meaning**: MULTI parameter is invalid. **Action**: Use MULTI= to specify the parameter; check your program for incorrect modifying of the parameter list.	EDG3909E
108	56	**Meaning**: The token is already in use. **Action**: Use TOKEN=**token** to specify a token that is not in use.	EDG3910E
108	58	**Meaning**: OUTPUT=FIELDS is not supported for the subcommand specified by SUBCMDADDR=**subcmdaddr**. **Action**: Use OUTPUT=LINES or specify a different subcommand.	EDG3911E
108	60	**Meaning**: The length of the subcommand specified by SUBCMDADDR=**subcmdaddr** is too large. **Action**: Use a smaller subcommand.	EDG3912E
112	—	**Meaning**: Environmental error. A limit, such as a storage limit, was exceeded. The operation you requested was not successfully completed. **Action**: Refer to the action provided with the specific reason code.	
112	02	**Meaning**: Unable to obtain sufficient work area storage. **Action**: Remove the cause of the short-on-storage condition or request a larger region size. Rerun your program.	EDG3913E
116	—	**Meaning**: System error. An error caused by the system, rather than your program, has been encountered. The operation you requested was not successfully completed. **Action**: Refer to the action provided with the specific reason code.	

Table 4. Return and reason codes for the EDGXCI macro (continued)			
Return Code	**Reason Code**	**Meaning and Action**	**Related Message**
116	02	**Meaning**: DFSMSrmm is not installed. **Action**: Ensure DFSMSrmm is installed and active before running your program.	EDG3914E
116	04	**Meaning**: A call to a system service has resulted in a non-zero return code. DFSMSrmm has placed the return code and the associated reason code as structured fields in your output buffer. **Action**: Retry the subcommand after the cause of the error has been corrected or removed.	EDG3915E
116	06	**Meaning**: An abnormal end has occurred. **Action**: Remove the cause of the abnormal end. Rerun your program.	EDG3916E
120	02	**Meaning**: Program error has occurred while you were using the high-level API. **Action**: Refer to the action provided with the specific reason code.	EDG3918E
120	04	**Meaning**: The LOAD for program EDGXAPI failed. **Action**: Correct the cause of the error and retry the command.	EDG3919E

EDGXCI example

You can modify the example shown here to:

- Obtain space for your output buffer in your work area in dynamic storage.
- Obtain space for the parameter list in your work area in dynamic storage.
- Specify subcommands that have this format:
  - The subcommand is prefixed by a two-byte length.
  - The subcommand is specified as a single input string.
- Use addresses that are pointer fields.
- Reuse the same parameter list for many requests.
- Reuse your 4-byte token area by specifying TOKEN= on all EXECUTE forms of EDGXCI. Your 4-byte token area is updated on return from the DFSMSrmm API.
- Make the list form parameter list large enough for all the parameters you might specify by using PLISTVER=MAX on the execute form of the EDGXCI macro.

Note: SAMPLIB member EDGAPISR provides a similar example of using EDGXCI.

Macro continuation characters must be entered in column 72.

```
YOURPGM  CSECT
R0       EQU   0
R1       EQU   1
R3       EQU   3
R4       EQU   4
R9       EQU   9
R11      EQU   11
R12      EQU   12
```

```
R13      EQU   13
R15      EQU   15
*        ..
         USING *,R11
         USING WORKDS,R12
         LA    R13,REGSAVE        Point to register save area
*        ..
*        ..
         LA    R0,OUTBUFWK        Save the
         ST    R0,APIOUTB@           address of output buffer
*************************************************************
*        Load the API module                                **
*************************************************************
         LOAD  EP=EDGXAPI
         ST    R0,APIMOD@         Save API module address
*        ..
         XC    MYTOKEN,MYTOKEN    Ensure no token yet
         LA    R4,LISTV@          List volume subcmd address
         BAL   R9,BEGINCMD        Begin the command
*        ..
*************************************************************
*        Going to reuse the resources, instead of releasing**
*        resources obtained by the API for the 1st BEGIN   **
*************************************************************
         LA    R4,SEARCHD@        Search subcmd address
         BAL   R9,BEGINCMD        Begin the command
*        ..
         BAL   R9,MOREDATA        Get more data for search
*        ..
         BAL   R9,RELEASE         All done, release resources
*        ..
*************************************************************
*        Delete the API module                              **
*************************************************************
         DELETE EP=EDGXAPI
*        ..
*************************************************************
**       Call API to begin a new subcommand                 **
*************************************************************
BEGINCMD DS    0H
CALL1    EDGXCI  MF=(E,MYPL),PLISTVER=MAX,                  X
               APIADDR=APIMOD@,OPERATION=BEGIN,             X
               TOKEN=MYTOKEN,                               X
               SUBCMDADDR=(R4),OUTBUFADDR=APIOUTB@
         BR    R9                 Return
*************************************************************
**       Call API to get more data for current subcommand  **
*************************************************************
MOREDATA DS    0H
CALL2    EDGXCI  MF=(E,MYPL,NOCHECK),PLISTVER=MAX,          X
               OPERATION=CONTINUE,TOKEN=MYTOKEN
         BR    R9                 Return
*************************************************************
**       Call API to release resource such as storage and  **
**            loaded modules.                               **
*************************************************************
RELEASE  DS    0H
REL1     EDGXCI  MF=(E,MYPL,NOCHECK),PLISTVER=MAX,          X
               OPERATION=RELEASE,TOKEN=MYTOKEN
         BR    R9                 Return
*************************************************************
**       SEARCH DATA SET SUBCOMMAND                         **
*************************************************************
SEARCHD  DS    0C
         DC    AL2(SEARCHDL)
         DC    C'SEARCHDATASET ....'
SEARCHDL EQU   *-SEARCHD
SEARCHD@ DC    A(SEARCHD)
*************************************************************
**       LISTVOLUME SUBCOMMAND                              **
*************************************************************
LISTV    DS    0C                 Listv command buffer
         DC    AL2(LISTVL)        Length of command
         DC    C'LISTVOLUME ....'
LISTVL   EQU   *-LISTV            Length of command
LISTV@   DC    A(LISTV)           Address of command
*        ..
*************************************************************
**       PROGRAM WORK AREA                                  **
*************************************************************
WORKDS   DSECT
APIOUTB@ DS    A                  Pointer to output buffer
```

```
APIMOD@  DS    A                 Address of the API module
REGSAVE  DS    18F               Save area
MYTOKEN  DS    CL4               Token from the API
*************************************************************
**       PARAMETER LIST DEFINITION                         **
*************************************************************
         EDGXCI MF=(L,MYPL,0D),PLISTVER=MAX PLIST area
         DS    0D
OUTBUFWK DS    CL4096            Output buffer area
*************************************************************
**       STRUCTURED FIELD DEFINITIONS                      **
*************************************************************
SFDEFDS  DSECT
         EDGXSF
         END
```

Chapter 2. Using the object-oriented DFSMSrmm application programming interface using C++

DFSMSrmm samples provided in SAMPLIB: EDGHCLT is shipped in SAMPLIB. The sample code shows how to issue RMM subcommands by using the DFSMSrmm high-level language application programming interface classes and methods.

Requirement: The dynamic link library (DLL) is compiled using the IBM z/OS V1R10 XL C/C++ compiler. To compile your own program, you can use compiler versions up to and including the IBM z/OS V1R10 XL (ISO C/C++) level of the compiler.

Related reading: For information about using the IBM z/OS V1R10 XL C/C++ compiler, see *z/OS XL C/C++ User's Guide*. For migration and compatibility considerations, see *z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer*.

You can use C++ and other high-level programming languages to write programs to obtain information about DFSMSrmm resources. You use the same DFSMSrmm subcommand strings that you can use with the EDGXCI application programming interface. You can get output as structured field introducers or in Extensible Markup Language (XML). The XML output contains data and tags to define the data. DFSMSrmm provides a schema called rmmxml.xsd that contains the definitions for the XML. For XML output, DFSMSrmm converts the data to character in Unicode format as defined in the XML Schema file for the DFSMSrmm resources. See “Receiving extensible markup language (XML) output data in the XML output buffer” on page 21.

To create your own program as shown in Figure 2 on page 16, you need access to the EDGXHCLU (header file) and the EDGXHCLL (definition side deck). The header file is necessary for the compile step and located in SYS1.MACLIB. The definition side deck is necessary for the bind step and is located in SYS1.SIEASID.

```
//COMPBIND  JOB (4378),'COMPILE BIND HCLT',MSGCLASS=H,MSGLEVEL=(1,1),
//         TIME=3,CLASS=A,REGION=0M,NOTIFY=&SYSUID
//*
//*********************************************************************
//*                                                                   ***
//*  COMPILE AND BIND A C++ API USERPROGRAM                           ***
//*                                                                   ***
//*********************************************************************
//*
//*********************************************************************
//* COMPILE STEP:                                                      *
//* SYSLIB : LIBRARIES for C++ CLASS DEFINITION FILES AS SOURCE CODE   *
//*          INCLUDED IN THE USER PROGRAM, RMM HLL API CLASS           *
//*          DEFINITION FILE IS EDGXHCLU IN SYS1.MACLIB                *
//*********************************************************************
//COMPILE  EXEC PGM=CCNDRVR,
//         PARM=('/CXX OPTFILE(DD:CPARMS)'),
//         REGION=80M
//CPARMS   DD  *
           XREF,OPTIMIZE,SOURCE,OBJ,MAR,
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//         DD  DSN=CEE.SCEEH.H,DISP=SHR
//         DD  DSN=CEE.SCEEH.SYS.H,DISP=SHR
//SYSLIN   DD  DSN=&SYSUID..CPP.OBJ(EDGHCLT),DISP=SHR
//SYSIN    DD  DSN=&SYSUID..CPP.SOURCE(EDGHCLT),DISP=SHR
//*
//*********************************************************************
//* BIND STEP:                                                         *
//* COMPILED MODULE EDGHCLT NEEDS TO BE CONCATENATED WITH DEFINITION *
//* SIDE DECK : SYS1.SIEASID(EDGXHCLL) SAME MEMBER NAME AS DLL        *
//*                                                                    *
//* SYSLMOD : OUTPUT DATASET (HLQ.CPP.LOAD)HAS TO BE PDSE FORMAT   *
//*********************************************************************
//BINDCPP  EXEC PGM=IEWL,REGION=1024K,
//         PARM='AMODE=31,MAP,RENT,DYNAM=DLL'
//SYSLIB   DD  DISP=SHR,DSN=CEE.SCEECPP
//         DD  DISP=SHR,DSN=CEE.SCEELKED
//SYSLMOD  DD  DISP=SHR,DSN=HLQ.CPP.LOAD
//SYSLIN   DD  DISP=SHR,DSN=&SYSUID..CPP.OBJ(EDGHCLT)
//         DD  DISP=SHR,DSN=SYS1.SIEASID(EDGXHCLL)
//         DD  DDNAME=SYSIN
//SYSDEFSD DD  DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
    NAME EDGHCLT(R)  RC=0
//*
```

Figure 2. Sample job control language (JCL) for prelink step

Figure 3 on page 16 shows sample JCL that you can use to request information for the RMM LISTVOLUME subcommand.

```
//*-------------------------------------------------------------------*
//* JCL Example to use C/C++ HLLAPI submitting RMM LIST VOLUME command,*
//* using sample program EDGHCLT,                                     *
//* EDGHCLT needs access to DLL: SYS1.SIEALNKE(EDGXHCLL)              *
//* receiving SFI output (SFIFILE) and XML output (XMLFILE)           *
//*-------------------------------------------------------------------*
//SMPLAPI  EXEC PGM=EDGHCLT,PARM='"LISTVOLUME A00001"'
//STEPLIB  DD DISP=SHR,DSN=HLQ.CPP.LOAD
//XMLFILE  DD DISP=(NEW,CATLG),DSN=USERID.OUTPUT.XMLFILE,
//            UNIT=SYSALLDA,VOL=SER=RMMDSK,
//            SPACE=(CYL,(5,5)),DCB=(RECFM=VB,LRECL=1028,BLKSIZE=6144)
//SFIFILE  DD DISP=(NEW,CATLG),DSN=USERID.OUTPUT.SFIFILE,
//            UNIT=SYSALLDA,VOL=SER=RMMDSK,
//            SPACE=(CYL,(5,5)),DCB=(RECFM=VB,LRECL=1028,BLKSIZE=6144)
//SYSPRINT DD SYSOUT=*
```

Figure 3. Sample JCL for requesting LISTVOLUME information

You need to write the program using C++ using the DFSMSrmm API classes and DFSMSrmm API methods to establish the connection to DFSMSrmm, issue the DFSMSrmm subcommands, and receive the output. If you select SFI format for the output, DFSMSrmm returns the information in structured field formats with all the fields provided.

Here is sample code that you can modify to use the high-level application programming interface.

```
/*********************************************************************
*                                                                    *
*  Module Name:  EDGHCLT                                             *
*                                                                    *
*  Description:  SAMPLE CODE for USING C/C++ HIGH LEVEL API INTERFACE *
*                                                                    *
*********************************************************************
*                                                                    *
* z/OS DFSMSrmm V1R11                                                *
*                                                                    *
* PROPRIETARY V3 STATEMENT                                           *
* Licensed Materials - Property of IBM                               *
* 5694-A01                                                           *
* Copyright IBM Corp. 1993,2009                                      *
* END PROPRIETARY V3 STATEMENT                                       *
*********************************************************************
*                                                                    *
*  Function:                                                         *
*                                                                    *
*    This C++ Module is a sample program for the customer to use     *
*    the High Level Language C/C++ API                               *
*                                                                    *
*********************************************************************
*                                                                    *
* Change History                                                     *
*                                                                    *
* $LV=RMMV1R6,1R6,030707 BRB: Created High Level API Interface    @LVA*
*                                                                    *
*********************************************************************/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream.h>
#include "EDGXHCLU"

FILE* sfiFp;

/*****************************************************************
 *  function to print SFI buffer into file                       *
 *****************************************************************/
 void printSFItoFile(RmmInterface::t_outp* outputPtr)
 {
   int outputlen=outputPtr->header.out_used;
   char* p = outputPtr->outputBuffer;
   char  ch;
   int i,len = 0;
   int offset = 0;
   int l = 0;

   for (l=0; l < outputlen; l++)
   {
     len = (*p * 16) + *(p+1);

     if ( len == 0 ) break;

     fwrite(p,1,len,sfiFp);

     p = p + len;
   }

 }
/*****************************************************************
* start main                                                     *
*****************************************************************/
int main(int argc, char* argv [])
{
  long  rc = 0;
  FILE* xmlFp;
  RmmApi* pApi;
  RmmCommand* pCom;
  char* tsoCommand;
  tsoCommand = argv[1];

/*****************************************************************
* get Output File names and open files                           *
*****************************************************************/

  if ( (xmlFp = fopen("DD:XMLFILE","w")) == NULL )
  {
```

```
     printf("could not open %s\n","DD:XMLFILE");
     exit(0);
  }
  if ( (sfiFp = fopen("DD:SFIFILE","wb,type=record")) == NULL )
  {
     printf("could not open %s\n","DD:SFIFILE");
     exit(0);
  }

/*************************************************************
* create RmmApi object                                       *
*************************************************************/
  pApi = new RmmApi();
  printf(" \nAPI object created \n");

/*************************************************************
* open Api                                                   *
*************************************************************/
  if ( pApi->openApi() == 0 )
    {
     printf("API Return Code : %d\n",pApi->getApiRC());
     printf("API Reason Code : %d\n",pApi->getApiRS());
     printf("API Message     : %s\n",pApi->getMessageText());
    }
  else
    {
     printf("Could not open API \n");
     exit(0);
    }
/*************************************************************
* create RmmCommand object                                   *
*************************************************************/

     pCom = new RmmCommand(pApi);
/*************************************************************
* processes a TSO command                                    *
*************************************************************/

   rc = pCom->issueCmd(tsoCommand);

   switch ( rc )
   {
     case 0 :
        printf("Return Code : %d\n",pCom->getApiRC());
        printf("Reason Code : %d\n",pCom->getApiRS());
        printf("Message     : %s\n",pCom->getMessageText());
        printSFItoFile((RmmInterface::t_outp*) pCom->getBufferSfi());
        fprintf(xmlFp,"%s\n",pCom->getBufferXml());
        break;

      case 1 :
        printf("Return Code : %d\n",pCom->getApiRC());
        printf("Reason Code : %d\n",pCom->getApiRS());
        printf("Message     : %s\n",pCom->getMessageText());
        printSFItoFile((RmmInterface::t_outp*) pCom->getBufferSfi());
        fprintf(xmlFp,"%s\n",pCom->getBufferXml());

        while( (pCom->getApiRC()==0) && (pCom->getApiRS()==4) )
        {
          rc = pCom->getNextEntry();
          printf("Return Code : %d\n",pCom->getApiRC());
          printf("Reason Code : %d\n",pCom->getApiRS());
          printf("Message     : %s\n",pCom->getMessageText());
          printSFItoFile((RmmInterface::t_outp*) pCom->getBufferSfi());
          fprintf(xmlFp,"%s\n",pCom->getBufferXml());
        }
        break;

      case -1:
        printf("Return Code : %d\n",pCom->getApiRC());
        printf("Reason Code : %d\n",pCom->getApiRS());
        printf("Message     : %s\n",pCom->getMessageText());
        break;

      default:
        printf("Return Code : %d\n",pCom->getApiRC());
        printf("Reason Code : %d\n",pCom->getApiRS());
        printf("Message     : %s\n",pCom->getMessageText());
   }

/*************************************************************
* destruction
```

```
************************************************************/
  delete pCom;
  delete pApi;

  fclose(sfiFp);
  fclose(xmlFp);
  exit(0);
}                              /* end main */
```

DFSMSrmm high level language API classes

C++ classes

Use the DFSMSrmm RmmApi class to prepare the environment for using the RmmCommand class to use the DFSMSrmm TSO subcommands with the API. You can also use the RmmTransaction class that makes use of the RmmApi and RmmCommand classes. All of these classes are defined in the DFSMSrmm header file EDGXHCLU.

Table 5. DFSMSrmm API command C++ classes

Class	Description
RmmInterface	This is the superclass for DFSMSrmm processing. This class provides methods that are common to the classes RmmApi and RmmCommand. This class cannot be instantiated.
RmmApi	This class extends the RmmInterface class. Use this class to create an object to initiate a communication session with DFSMSrmm. You must create an instance of this class before you use class RmmCommand. This instance can be used to create one or more RmmCommand objects to enable you to run DFSMSrmm subcommands. You need one RmmApi object for each Multiple Virtual Storage (MVS™) TCB under which DFSMSrmm runs. To end the communication session with DFSMSrmm and to no longer run subcommands, delete the RmmApi object.
RmmCommand	This class extends the RmmInterface class. Use this class to process a DFSMSrmm TSO subcommand. You must pass a reference to the RmmApi object when you instantiate an instance of this class. You can instantiate multiple instances of the RmmCommand class to process multiple commands in parallel. For example, you can use the output from a SEARCH command to issue LIST subcommands.
RmmTransaction	This class makes use of the RmmApi and RmmCommand classes. Instantiate an instance of this class, if you want to use the runCommandXml method.

Java class

If you want a Java application to access DFSMSrmm, use class RmmJApi.

Table 6. DFSMSrmm API command Java class

Class	Description
RmmJApi	Instantiate an instance of this class to communicate with DFSMSrmm from a Java application.

DFSMSrmm API methods

Use the DFSMSrmm API methods to retrieve and update information about DFSMSrmm-managed resources. The naming convention for the methods is ClassName.methodName.

Table 7. DFSMSrmm API C++ methods

Method	Description
RmmApi.openApi()	Use this method to check that DFSMSrmm is active and available to process commands.
RmmApi.closeApi()	Use this method when you no longer want to communicate with DFSMSrmm using this command session.
RmmCommand.issueCmd()	Use this method to issue a subcommand to DFSMSrmm. DFSMSrmm returns the subcommand return code and reason code. To access the output from the subcommand, use the getBufferSfi method or the getBufferXml method.

Table 7. DFSMSrmm API C++ methods (continued)

Method	Description
RmmCommand.getBufferSfi()	Use this method to obtain a string that contains the SFI output buffer from subcommand processing. Use this method after using the RmmCommand.issueCmd method and after using the RmmCommand.getNextEntry method.
RmmCommand.getBufferXml()	Use this method to obtain a string that contains the XML output converted from the SFI output of subcommand processing.
RmmCommand.getNextEntry()	Use this method to retrieve information for the next resource or set of resources when there is more than one resource to be returned. For example, SEARCH subcommands and LISTCONTROL subcommands can return more than one resource. The getBufferXml and getBufferSfi methods can return multiple resources in a buffer; be sure to process all the returned data (XML or SFIs) before using the getNextEntry method if more entries may exist.
RmmInterface.getMessageText()	Use this method to obtain a string that contains the DFSMSrmm information or error message for the last command issued or the last getNextEntry method processing.
RmmInterface.getApiRc()	Use this method to obtain the return code from the last API request. Use the getMessageText method to retrieve the corresponding information or error message. See "EDGXCI return and reason codes" on page 9 for information about message processing.
RmmInterface.getApiRs()	Use this method to obtain the reason code from the last API request. Use the getMessageText method to retrieve the corresponding information or error message. See "EDGXCI return and reason codes" on page 9 for information about message processing.
RmmTransaction.runCommandXml()	Use this method to return a string containing the XML output converted from the SFI output of subcommand processing. It may also return error messages and return and reason codes for the command in the XML.
RmmTransaction.runCommandXmlShort()	Use this method to return a string containing the XML output for key values only. Only specific search commands return key fields. For example: • For SearchVolume, only the volser is returned. • For SearchDataset, only the datasetname, volume, and filesequence number are returned. • For SearchOwner, only the owner ID is returned. • For SearchRack/SearchBin, only the rack/bin number, location, and media name are returned. Other commands work as well, but they return all of the data, not just the key values.

Java methods

Table 8. DFSMSrmm API Java methods

Method	Description
RmmJApi.runCommandXml()	Use this method to return a string containing the XML output converted from the SFI output of subcommand processing. It may also return error messages and return and reason codes for the command in the XML.

Table 8. DFSMSrmm API Java methods (continued)

Method	Description
RmmJApi.runCommandXmlShort()	Use this method to return a string containing the XML output for key values only. Only specific search commands return key fields. For example: • For SearchVolume, only the volser is returned. • For SearchDataset, only the datasetname, volume, and filesequence number are returned. • For SearchOwner, only the owner ID is returned. • For SearchRack/SearchBin, only the rack/bin number, location, and media name are returned. Other commands work as well, but they return all of the data, not just the key values.

Receiving extensible markup language (XML) output data in the XML output buffer

Use the high-level language application programming interface to obtain output in XML format. The XML output might also return error messages and return and reason codes.

Figure 4 on page 21 shows an example that issues an RMM SEARCHRACK subcommand and writes the XML output into the file named XMLFILE.

You can work with the output data in XML format by writing the output into a file or by parsing the output directly. You can define this file in the JCL, which you use to issue the command.

This example shows in C++ code how to:

- Issue a DFSMSrmm TSO subcommand by using the method issueCommand() .
- Use the method getBufferXml() to obtain access to the XML data.

```
FILE*  xmlFp;                                  /* declare file pointer */
RmmApi* pApi;                                  /* declare an Api object */
RmmCommand* pCom;                              /* declare a Command object */
pApi = new RmmApi();                           /* create an Api object */
pApi->openApi();                               /* open Api */
pCom = new RmmCommand(pApi);                   /* create a Command object */
pCom->issueCmd("SR RACK(*)");                  /* issue a Command */
xmlFp = fopen("DD:XMLFILE","w")                /* open the file for writing */
fprintf(xmlFp,"%s",pCom->getBufferXml());      /* print the data into the file */
fclose(xmlFp);                                 /* close the file */
```

Figure 4. C++ code example for writing XML output to a file

Figure 5 on page 21 shows the content of the file XMLFILE.

```
<?xml version="1.0" encoding="EBCDIC-CP-US" ?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="/usr/lib/xml_schema/rmmxml.xsd">
<RACK>
<RCK>RACK  </RCK>
<VOL  xsi:nil="true"></VOL>
<RST>EMPTY</RST>
<LOC>SHELF</LOC>
<MEDN>3480</MEDN>
<PID>*</PID>
</RACK>
<INFO>
<RTNC>4</RTNC>
<RSNC>4</RSNC>
<MSGT>EDG3011I 1 ENTRY LISTED  </MSGT>
</INFO>
</document>
```

Figure 5. XMLFILE output file

Most of the DFSMSrmm-produced XML tags use the SFI names that are described in Table 16 on page 67. For example, the XML tag for volume is <VOL>, which corresponds to the SFI name VOL. The DFSMSrmm-produced XML tags that do not use the SFI names are these tags.

- The XML tag <VOLINFO> for the volume resource group.
- The XML tag <VRSINFO > for the VRS resource group.
- The XML tags <JBN2>, <NME2>, <SCD2>, and <SCN2>, which represent the structured field introducers <2JBN>, <2NME>, <2SCD> and <2SCN>. XML does not allow tags to start with numeric characters.
- The XML tags <DSS6> and <USE6> are structured using additional tags for factor (<*xxxx*F>) and value (<*xxxx*S>), where *xxxx* is the XML tag name.

The XML output structure is declared in the XML schema file RMMXML.XSD, that you find in your file system directory /usr/lib/xml_schema. The schema contains type definitions for all elements.

The XML data stream contains a Uniform Resource Identifier (URI) to reference the required schema. To change the schema location, use the XML parser setExternalnoNamespaceSchemaLocation method.

DFSMSrmm ensures it creates only well-formed and valid XML documents and ensures that any text within an element contains only valid characters. The special characters &, <, >, ", and ' are escaped using the entities:

&
    &

<
    <

>
    >

"
    "

'
    '

Your XML parser converts the entities back to the correct text character. Any code that processes the XML document without a parser must consider that these entities might exist within the document and should be converted back to the correct character before use of the data.

Related reading: You can write your own application to parse the XML data by using the XML parser. IBM provides an XML parser and sample applications in the XML Toolkit for z/OS (www.ibm.com/systems/z/os/zos/tools/xml) or from the IBM Software Delivery for System Modification Program Extended (SMP/E) installation.

Chapter 3. Using the DFSMSrmm application programming interface using assembler language

Use the general programming guidelines to help you write your application program.

Obtaining resources

When you begin a new subcommand request and provide a token that is set to all zeros, the DFSMSrmm API obtains a new set of resources. When you begin a new subcommand request and reuse a valid, nonzero token, DFSMSrmm reuses resources associated with the token.

To use resources most efficiently, consider these items.

- Use a different output buffer for each RMM TSO subcommand request. Reuse an output buffer to begin a new subcommand request only when there is nothing in the buffer that you need.
- Allocate a sufficient number of token areas, and parameters lists.
- Use the correct token when continuing a RMM TSO subcommand or when releasing a particular set of resources.
- Reuse a token to begin a new RMM TSO subcommand only when you no longer need the information obtained from the previous request.
- Reuse the resources associated with the token, especially when you are processing hundreds or thousands of subcommands.

Specifying TSO subcommand input in the EDGXCI macro

To obtain information from the DFSMSrmm control data set, specify a DFSMSrmm TSO subcommand as a single input line without the RMM command, as shown in Figure 6 on page 23.

```
AV MLV001 STATUS(MASTER) EXPDT(98001) OWNER(IBMUSER) OWNERACCESS(UPDATE) RACK(ML0001)
```

Figure 6. Example of specifying the DFSMSrmm API subcommand

Do not specify it as an RMM command with multiple input lines, as shown in Figure 7 on page 23.

```
RMM  AV MLV001 STATUS(MASTER) EXPDT(98001) OWNER(IBMUSER)-
        OWNERACCESS(UPDATE) RACK(ML0001)
```

Figure 7. Example of specifying the RMM TSO subcommand

In addition, specify subcommands using fully specified subcommand operands and their values. Avoid abbreviating the subcommands or operands because they can change when new subcommand operands and values are added.

Using the CONTINUE operation in the EDGXCI macro

Use the EDGXCI OPERATION=CONTINUE parameter in your application program to ensure that you obtain all the available data. When you use OPERATION=CONTINUE, you might not receive more output data or you might receive only messages in your output buffer.

The DFSMSrmm API can return control back to your application program before returning all the data you expect because:

- There is no more room in the output buffer for the additional data.
- The API stops after returning data for a single resource when you issue a request that uses a SEARCH command with OUTPUT=FIELDS and MULTI=NO is specified (or assumed by default).

- There is no more data to return to your application program.

The DFSMSrmm API issues return codes and reason codes indicating the results of processing when you specify OPERATION=CONTINUE. Write your application program to check the return codes and reason codes that the DFSMSrmm API returns to your application program.

Table 9. Return codes and reason codes issued when you specify OPERATION=CONTINUE

Return Code	Reason Code	Processing
0	0	DFSMSrmm issues this return code and reason code in response to a search type subcommand. DFSMSrmm will not return any more records because there are no more records to return or because the search limit has been reached.
0	4	DFSMSrmm issues this return code and reason code when you issue requests specifying the LISTCONTROL subcommand and there are more records to return. Specify the OPERATION=CONTINUE to obtain more records.
4	2	DFSMSrmm issues this return code and reason code in response to a SEARCH type subcommand. The DFSMSrmm API issues these codes when the search limit you set for a DFSMSrmm subcommand has been reached but there might be more records to return.
4	4	DFSMSrmm issues this return code and reason code in response to a search type subcommand. The DFSMSrmm API issues these codes when the search processing indicates fewer records returned than were requested.
4	8	DFSMSrmm issues this return code and reason code in response to a search type subcommand. The DFSMSrmm API issues these codes when no entry meets then search criteria during search processing.

See “Controlling output from list and search type requests” on page 58 for an example of the interaction between the size of an output buffer, the amount of output data the API returns, and the LIMIT value you set.

Requesting multiple resources for SEARCH subcommands

The DFSMSrmm API can return resources either one at a time or multiple at a time when you specify one of the DFSMSrmm TSO RMM SEARCHDATASET, SEARCHBIN, SEARCHOWNER, SEARCHPRODUCT, SEARCHRACK, SEARCHVOLUME, and SEARCHVRS subcommands together with OUTPUT=FIELDS. Use the MULTI keyword to notify the API about which type of output you can handle. To specify MULTI=YES, your application must be able to handle multiple resources each separated by the begin/end group structured field introducers. When you specify MULTI=YES, your output buffer can have one or more resource groups returned in a single call of the API. Using MULTI=YES helps reduce the system resources used for API processing.

Using parameter lists to pass information to the DFSMSrmm API

You can write your application program to include this processing:

- Serially or concurrently process subcommands.
- Use single parameter lists or multiple parameter lists for each subcommand. For example, your application program can use one parameter list for a SEARCH type of subcommand and another parameter list for a CHANGE type of subcommand.
- Reuse resources (tokens).

You can use variations of parameter lists and tokens in your application program to meet your application requirements.

Table 10. Types of parameter lists

Variation	Guidelines	Reference
Single parameter list and a single token area	• Only one subcommand request can be active at a time. • An active subcommand request must be completed before beginning another subcommand request.	"Coding a single parameter list, single token area" on page 25
Single parameter list and multiple token area	• More than one subcommand request can be active at a time. • Only one subcommand request can be processed at any given time.	"Coding a single parameter list, multiple token areas" on page 26
Multiple parameter lists with a single token area	• Only one subcommand can be active at a time. • Different parameter lists can be used for these tasks: – Begin subcommand requests. – Continue subcommand requests. – Release resources. • Starting a new subcommand request ends any previous subcommand request.	"Coding multiple parameter lists, single token area" on page 28.
Multiple parameter lists and multiple token area	• More than one subcommand request can be active at a time. • More than one active subcommand request can be processed at a time. • Different parameter lists can be used to: – Begin subcommand requests. – Continue subcommand requests. – Release resources.	"Coding multiple parameter lists, multiple token areas" on page 29

For illustrative purposes, the examples use inline code segments with shortened code lines.

Coding a single parameter list, single token area

Figure 8 on page 26 is an example of how your application program can use a single parameter list and a single token area. The example includes a BEGIN, CONTINUE, and RELEASE for each subcommand request because you are not reusing resources. You need a new token for the second subcommand request because you are not reusing any resources and need a separate token for each request.

```
*****************************************
** Start the first subcommand
*****************************************
  XC    TOKENA,TOKENA           No resources/token yet
  LA    R4,SUBCMD1              Point to 1st subcommand
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKENA,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
  ...*****************************************
** Continue the subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKENA,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Done with the subcommand, release
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKENA
  ...
*****************************************
** Start the second subcommand
*****************************************
  LA    R4,SUBCMD2              Point to 2nd subcommand
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKENA,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
  ...
*****************************************
** Continue the subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKENA,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Done with the subcommand, release
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKENA
```

Figure 8. Single parameter list, single token area

The example includes the OPERATION=RELEASE parameter. When you use OPERATION=RELEASE, DFSMSrmm releases work areas that contain data and pointers for the subcommand. You must obtain resources for the next subcommand request. You might improve performance by deleting the OPERATION=RELEASE for the first subcommand. Then when you begin the second subcommand, the DFSMSrmm API module reuses resources, such as work areas, that it obtained for the first subcommand. Reusing resources can reduce processing overhead associated with releasing and obtaining resources.

If you do not use OPERATION=RELEASE, when the second subcommand request starts, all data and pointers for the first subcommand are overwritten.

For OPERATION=RELEASE, you do not specify SUBCMDADDR or OUTBUFADDR. For OPERATION=CONTINUE, you do not specify SUBCMDADDR.

Coding a single parameter list, multiple token areas

This variation allows you to continue a previous subcommand after you have started another. You might need to use multiple token areas when your application program is designed to support a sequence of subcommand requests like the one that follows:

1. Use a SEARCHVOLUME subcommand to request volume information. For example:

```
SEARCHVOLUME OWNER(userid) LIMIT(*)
```

2. Use a SEARCHDATASET subcommand to obtain data set information. For example:

```
SEARCHDATASET VOLUME(volser) LIMIT(*)
```

3. Repeat subcommands until all information for all data sets is obtained and passed back to your user.

Figure 9 on page 27 shows how you can use a single parameter list and multiple tokens to identify work areas. The multiple token areas allow the flexibility of continuing a previous subcommand after starting another subcommand. Use the token you obtained from the previous subcommand when you want to continue that subcommand.

```
*****************************************
** Start the first subcommand
*****************************************
  XC    TOKEN1,TOKEN1            No resources/token yet
  LA    R4,SUBCMD1               Point to 1st subcommand
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKEN1,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
  ...
*****************************************
** Start the second subcommand
*****************************************
  XC    TOKEN2,TOKEN2            No resources/token yet
  LA    R4,SUBCMD2               Point to 2nd subcommand
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKEN2,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
  ...
*****************************************
** Continue the second subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKEN2,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Continue the first subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKEN1,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Release resources for the first subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKEN1
  ...
*****************************************
** Release resources for the second subcommand
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKEN2
```

Figure 9. Single parameter list, multiple token areas

Figure 9 on page 27 shows how you can reuse resources. When your application program is finished with the first subcommand request, it can reuse the first token to begin a third request. When that token is reused to begin a new subcommand request, you cannot continue the previous request associated with that token.

In Figure 9 on page 27, the same output buffers are used for all subcommand requests. As a result, all of the output data in the output buffer must be processed before another request can be started or continued. To avoid this situation, you might write your application program to use multiple output buffers instead of a single output buffer.

Figure 9 on page 27 shows multiple releases using the OPERATION=RELEASE parameter. Instead of using multiple releases, you can specify the OPERATION=ENDALL once to free all resources associated with all tokens. See Figure 10 on page 28 for an example of this method.

Note: You do not specify the TOKEN parameter when you use OPERATION=ENDALL. Your application program, however, is responsible for setting all tokens to zeros to prevent them from being reused.

```
*****************************************
** Release all resources
*****************************************
  EDGXCI  MF=(E,PLIST),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=ENDALL
```

Figure 10. Releasing all resources

Your application program might encounter a resource constraint condition like short-on-storage before it issues the OPERATION=ENDALL.

Coding multiple parameter lists, single token area

Figure 11 on page 28 shows how you can use multiple parameter lists and a single token area. With a single token area, you cannot continue the first subcommand request, even though there are multiple parameter lists. The variation in Figure 11 on page 28 prevents you from continuing the first subcommand after you begin the second subcommand.

```
*****************************************
** Start the first subcommand
*****************************************
  XC    TOKENA,TOKENA           No resources/token yet
  LA    R4,SUBCMD1              Point to 1st subcommand
  EDGXCI  MF=(E,BEGINPL),PLISTVER=MAX,                  X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKENA,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
  ...
*****************************************
** Continue the subcommand
*****************************************
  EDGXCI  MF=(E,CONTPL),PLISTVER=MAX,                   X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKENA,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Done with the subcommand, release
*****************************************
  EDGXCI  MF=(E,RELPL),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKENA
  ...
*****************************************
** Start the second subcommand
*****************************************
  LA    R4,SUBCMD2              Point to 2nd subcommand
  EDGXCI  MF=(E,BEGINPL),PLISTVER=MAX,                  X
        APIADDR=APIMOD@,OPERATION=BEGIN,                X
        TOKEN=TOKENA,                                   X
        SUBCMDADDR=(R4),OUTBUFADDR=(R3)
*****************************************
** Continue the subcommand
*****************************************
  EDGXCI  MF=(E,CONTPL),PLISTVER=MAX,                   X
        APIADDR=APIMOD@,OPERATION=CONTINUE,             X
        TOKEN=TOKENA,                                   X
        OUTBUFADDR=(R3)
  ...
*****************************************
** Done with the subcommand, release
*****************************************
  EDGXCI  MF=(E,RELPL),PLISTVER=MAX,                    X
        APIADDR=APIMOD@,OPERATION=RELEASE,              X
        TOKEN=TOKENA
```

Figure 11. Multiple parameter lists, single token area

Coding multiple parameter lists, multiple token areas

This variation lends itself to processing in re-entrant code where subroutines can be created for commonly used code. Here is an example that shows how the same subroutines can be used to issue and process multiple subcommand requests with each having its own token and output buffer area.

```
        ****************************************
        ** Start the first subcommand
        ****************************************
          XC    TOKENA,TOKENA           No resources/token yet
          LA    R2,TOKENA               Point to 1st token
          LA    R3,OUTBUF1              Point to 1st buffer
          LA    R4,SUBCMD1              Point to 1st subcommand
          BAS   R9,BEGRTN               Issue command
          ...
        ****************************************
        ** Start the second subcommand
        ****************************************
          LA    R2,TOKENB               Point to 2nd token
          LA    R3,OUTBUF2              Point to 2nd buffer
          LA    R4,SUBCMD2              Point to 2nd subcommand
          BAS   R9,BEGRTN               Issue command
          ...
        ****************************************
        ** Continue the 2nd subcommand
        ****************************************
          LA    R2,TOKENB               Point to 2nd token
          BAS   R9,CONRTN               Continue 2nd cmd
          ...
        ****************************************
        ** Continue the 1st subcommand
        ****************************************
          LA    R2,TOKENA               Point to 1st token
          BAS   R9,CONRTN               Continue 1st cmd
          ...
        ****************************************
        ** Done with the subcommands, release
        ****************************************
          LA    R2,TOKENA               Point to 1st token
          BAS   R9,RELTRN               Release 1st token
          ...
          LA    R2,TOKENB               Point to 2nd token
          BAS   R9,RELTRN               Release 2nd token
          ...
BEGRTN    EQU   *
          EDGXCI  MF=(E,BEGINPL),PLISTVER=MAX,                  X
                APIADDR=APIMOD@,OPERATION=BEGIN,                X
                TOKEN=(R2),                                     X
                SUBCMDADDR=(R4),OUTBUFADDR=(R3)
          BR    R9
          ...
CONRTN    EQU   *
        ****************************************
        ** Continue the subcommand
        ****************************************
          EDGXCI  MF=(E,CONTPL),PLISTVER=MAX,                   X
                APIADDR=APIMOD@,OPERATION=CONTINUE,             X
                TOKEN=(R2),                                     X
                OUTBUFADDR=(R3)
          BR    R9
          ...
RELRTN    EQU   *
        ****************************************
        ** Done with the subcommand, release
        ****************************************
          EDGXCI  MF=(E,RELPL),PLISTVER=MAX,                    X
                APIADDR=APIMOD@,OPERATION=RELEASE,              X
                TOKEN=(R2)
          BR    R9
```

Specifying the option to free a resource

You can free a resource when you no longer need to use it by performing one of these actions:

- Use the OPERATION=RELEASE and TOKEN=*token* parameters to free all resources associated with the specified token as shown in Figure 12 on page 30.

```
************************************************
** Done with the subcommand, setup release parmlist
************************************************
  EDGXCI  MF=(M,RELPL,NOCHECK),PLISTVER=MAX,            X
        APIADDR=APIMOD@,OPERATION=RELEASE

************************************************
** Call the DFSMSrmm API
************************************************
  EDGXCI  MF=(E,RELPL,NOCHECK),TOKEN=TOKENA
```

Figure 12. TOKEN= specified on EDGXCI

Specifying TOKEN=TOKENA on the EXECUTE form of EDGXCI causes the 4-byte TOKENA area to be set to all zeros upon return from freeing the token.

TOKEN=*token* is required even when you specify MF=(E,label,NOCHECK), unless you also specify OPERATION=ENDALL. Specifying TOKEN=*token* causes the 4-byte token area to be updated upon return from the DFSMSrmm API. The token is set to all zeros by the EDGXCI macro expansion.

- Specify the OPERATION=ENDALL parameter to free all resources associated with all tokens, as shown in Figure 13 on page 30.

  Rule: You are responsible for setting applicable tokens to all zeros when you specify OPERATION=ENDALL.
- Your application program ends (end-of-task occurs).

Specifying the option to release a resource

To release a resource, you must have access to the tokens associated with the resources that you want to release. If you no longer have access to the tokens or you have set the tokens to all zeros before you use OPERATION=RELEASE, there are only two ways that resources can be freed:

- Your application program specifies OPERATION=ENDALL to free all resources associated with all tokens.
- Your application program ends (end-of-task occurs).

In Figure 13 on page 30, the OPERATION=ENDALL parameter is specified and TOKEN is not required.

```
************************************************
** Done with the subcommand, setup endall parmlist
************************************************
  EDGXCI  MF=(M,RELPL,NOCHECK),PLISTVER=MAX,            X
        APIADDR=APIMOD@

************************************************
** Call the DFSMSrmm API
************************************************
  EDGXCI  MF=(E,RELPL,NOCHECK),OPERATION=ENDALL
```

Figure 13. TOKEN= not specified on EDGXCI

Chapter 4. Using an alternative interface to the DFSMSrmm application programming interface

The EDGXHINT interface is an alternative interface to the DFSMSrmm application programming interface (API):

- Assembler or C/C++ programs can be linked together with module EDGXHINT to exploit the API interface provided.
- When using Java, you must use the Java Native Interface (JNI) to C/C++ before you can use EDGXHINT.

EDGXHINT is shipped as a load module in LINKLIB.

When using high level languages to write applications to obtain information about DFSMSrmm resources, you use the same DFSMSrmm subcommand strings that you can use with the EDGXCI interface. You get output as structured field introducers (SFIs). To receive output as an XML document, use the Object-Oriented DFSMSrmm Application Programming Interface Using C++.

Related reading:

1. *z/OS XL C/C++ User's Guide*
2. *Integrating Java with Existing Data and Applications on OS/390*, SG24-5142-00

To create a program exploiting the EDGXHINT interface, bind EDGXHINT together with your own module as shown in Figure 14 on page 31.

```
//BINDPGM  JOB (4378),'BIND A PROGRAM',MSGCLASS=H,MSGLEVEL=(1,1),
//         TIME=3,CLASS=A,REGION=0M,NOTIFY=&SYSUID
//*
//*********************************************************************
//*                                                                   ***
//*  BIND A C/C++ PROGRAM TO USE THE EDGXHINT INTERFACE TO RMM        ***
//*                                                                   ***
//*  SYSLMOD: OUTPUT DATASET (HLQ.CPP.LINKLIB) MUST BE PDSE FORMAT    ***
//*                                                                   ***
//*********************************************************************
//BIND EXEC PGM=IEWL,REGION=4M,
//    PARM='AMODE=31,MAP,RENT'
//SYSLIB   DD  DSN=CEE.SCEELKEX,DISP=SHR
//         DD  DSN=CEE.SCEELKED,DISP=SHR
//         DD  DSN=CEE.SCEECPP,DISP=SHR
//SYSLMOD  DD  DISP=SHR,DSN=HLQ.CPP.LOAD
//SYSPRINT DD  SYSOUT=*
//INOBJ    DD  DSN=HLQ.OBJ,DISP=SHR
//LINKLIB  DD  DISP=SHR,DSN=SYS2.LINKLIB
//SYSLIN   DD  *
    INCLUDE INOBJ(USERPROG)
    INCLUDE LINKLIB(EDGXHINT)
    NAME USERPROG(R)  RC=0
/*
```

Figure 14. Binding a C++ program for use of EDGXHINT

The application program must provide buffers for the:

- Command string you want to pass to the API
- Output you will receive back from the API. The minimum recommended size is 80KB. The larger the output buffer you provide, the more resources that can be returned by one call to EDGXHINT.
- Messages that may be issued by the API as result of your command. The minimum recommended size is 256 bytes

The application program also must fill an interface structure, which is used to communicate with the API. You can then call EDGXHINT by passing the pointer to the interface structure. For more details on the processing between your program and the RMM API, see Chapter 3, “Using the DFSMSrmm application programming interface using assembler language,” on page 23.

Parameter list to call EDGXHINT

Table 11. Parameter list for a call of EDGXHINT

Field	Description	Set from
Function code	1. Open API (start communication) 2. Close API (end communication) 3. Issue command (begin a request) 4. Get next buffer (continue a request) 5. Release (end a request)	User program
Pointer to the command buffer	The user program needs to obtain the storage for a buffer big enough to hold the TSO subcommand to be issued. Maximum is 255 byte. EDGXHINT will read the TSO command from this buffer.	User program
Pointer to the output buffer	The user program needs to obtain the storage for an output buffer. Minimum recommended is 80KB. EDGXHINT will use this buffer to return the data requested.	User program
Pointer to first message buffer	The user program must obtain the storage for a 256 byte buffer. This buffer should always be cleared before EDGXHINT is called, to delete pre-existing content. EDGXHINT will use this buffer to return a message resulting from the last issued command, if appropriate.	User program
Pointer to second message buffer	The user program must obtain the storage for a 256 byte buffer. This buffer should always be cleared before EDGXHINT is called, to delete pre-existing content. EDGXHINT will use this buffer to return a second message resulting from the last issued command, if appropriate.	User program
Message count	Number of messages returned by EDGXHINT	EDGXHINT
API address	Address of EDGXAPI. Set by OPEN function. Can be used to determine if the API is open. If not NULL, then API is open.	EDGXHINT
MTAB address	Address of the DFSMSrmm message table. Set by OPEN function, used by EDGXHINT internally.	EDGXHINT
CMSG address	Address of the DFSMSrmm message routine. Set by OPEN function, used by EDGXHINT internally.	EDGXHINT
Token	Token used by macro EDGXCI to identify the request. The token is created at BEGIN processing (function 3) and used by CONTINUE processing (function 4). The token is cleared (set to zero) by EDGXHINT after RELEASE processing (function 5).	EDGXHINT
Return code	API return code	EDGXHINT
Reason code	API reason code	EDGXHINT

Interface structure to pass the parameter list to EDGXHINT

In C/C++ programming language, a struct is used to pass the parameter list to EDGXHINT. Sample code for this purpose is shown in Figure 15 on page 33. In this sample, the interface structure itself is defined in t_interface. Additional structs are used to map the command buffer (t_comm) and the output buffer (t_outp).

```
typedef struct t_comm               // to map the output buffer
{
    short com_length;               // length of the command
    char  commandBuffer[255]; // storage to hold the command
};
t_comm  commandStrct;               // variable of type t_comm
t_comm* commPtr;                    // pointer to t_comm

typedef struct t_outph              // to map the command buffer header
{
    long  out_length;               // length of the output buffer
    long  out_needed;               // output buffer length needed
    long  out_used;                 // output buffer length used
};
t_outph  outputHeaderStrct;         // variable of type t_outph

typedef struct t_outp               // to map the output buffer
{
    t_outph header;                 // output buffer header
    char  outputBuffer[80000]; // storage to hold the output
};
t_outp  outputStrct;                // variable of type t_outp
t_outp* outputPtr;                  // pointer to t_outp

typedef struct t_interface          // to map the interface structure
{
    long     function;              // function code
    t_comm*  command_ptr;           // pointer to command buffer
    t_outp*  outputBuf_ptr;         // pointer to output buffer
    char*    messageBuf_ptr1;       // pointer to first message buffer
    char*    messageBuf_ptr2;       // pointer to second message buffer
    long     messageCount;          // number of messages returned
    void*    addr_XAPI;             // address of module EDGXAPI
    void*    addr_MTAB;             // address of module EDGMTAB
    void*    addr_CMSG;             // address of module EDGCMSG
    long     token;                 // token to identify the request
    long     returncode;            // API return code
    long     reasoncode;            // API reason code
};
t_interface interStruct;            // variable of type t_interface
t_interface* pI;                    // pointer to t_interface
```

Figure 15. C/C++ sample code for an interface struct

Communication with the API

Define the API

Define the EDGXHINT program interface to your program, together with the interface struct, using code such as:

```
extern  "C"  int  EDGXHINT(  t_interface*  );
```

Start API communication

To start API communication, first initialize all elements of the interface structure and clear the buffers you provide. You can then open a communication session with the API by setting the function code to 1 (=OPEN) and calling EDGXHINT, passing the pointer to the interface struct:

```
interStruct.function        = 1L;
EDGXHINT(pI);
```

You can use the return and reason code elements of the interface structure to determine whether the open process was successful:

```
if ( interStruct.returncode == 0L
  && interStruct.reasoncode == 0L  )
  …..                       // successfully opened the API session
else
  ….                        // error handling needed
```

If the open process is successful, EDGXHINT fills the elements of the interface structure as described in Table 11 on page 32.

Issue a request

If the open is successful, you can start a request session by issuing a TSO subcommand through the API. Sample code for this is shown in Figure 16 on page 34. Place the command string in the command buffer, initialize buffers, set function code to 3 (=BEGIN), and call EDGXHINT.

```
char  command[12]  =  "SV  OWNER(*)";          // define the command
strcpy(commandStrct.commandBuffer,command);    // fill the command buffer
commandStrct.com_length = strlen(command)+2;   // set the command length
                                               // command length + 2 byte length field
strcpy(outputStrct.outputBuffer,'\0');         // clear output buffer
outputStrct.header.out_used=0;
strcpy(interStruct.messageBuf_ptr1,'\0');      // clear message buffers
strcpy(interStruct.messageBuf_ptr2,'\0');
interStruct.function = 3L;                     // set function code
EDGXHINT(pI);                                  // call EDGXHINT
```

Figure 16. Issue a TSO subcommand using EDGXHINT

You can evaluate the return and reason code to determine whether the command was processed successfully. From the message count, you can determine whether there are messages available in the message buffers. You will find returned data in the output buffer. This data is in SFI format and can be processed as described in Chapter 5, “Processing the output data in the output buffer,” on page 35. If a search command was issued, you will find one or more complete resources in the output buffer.

EDGXHINT always uses the EDGXCI MULTI=YES keyword on behalf of its callers. Therefore, all callers must be updated, if necessary, to handle a buffer containing multiple resources. A caller requiring the return of just a single resource can use the LIMIT(1) operand on the SEARCH subcommand.

Continue a request

If more matching resources exist (returncode = 0, reasoncode = 4), you might want to continue the request session. Clear the buffers, set function code to 4 (=CONTINUE) and call EDGXHINT again. The next set of resources are returned to the output buffer.

End a request

To end the request session, release the corresponding token. Set function code to 5 (=RELEASE) and call EDGXHINT.

```
interStruct.function        = 5L;
EDGXHINT(pI);
```

End API communication

To end communication with the API, set the function code to 2 (=CLOSE) and call EDGXHINT, passing the pointer to the interface struct:

```
interStruct.function        = 2L;
EDGXHINT(pI);
```

Return and reason codes using EDGXHINT

When using interface EDGXHINT, you receive return and reason codes, as described in “EDGXCI return and reason codes” on page 9.

Chapter 5. Processing the output data in the output buffer

The DFSMSrmm application programming interface returns data in the output buffer you define. The data is in this format:

- A four-byte length field into which your application program sets the total size of the output buffer.
- A four-byte length field that is used by DFSMSrmm when your output buffer is too small.
- A four-byte length field that contains the total size of all the output including the bytes of the length field.
- Structured fields, which consist of structured field introducers (SFIs) and data.
  - A structured field introducer (SFI) is a structure that separates one line or field of output data from another. Structured field introducers are described in "Description of structured fields" on page 35.
  - Data in line format or field format.

Use the EDGXSF macro described in "EDGXSF: Structured field definitions" on page 91 to map the output buffer header and the structured field introducers. EDGXSF also defines values used in the output fields. Do not hardcode the offsets because they might change in the future.

The DFSMSrmm API returns various types of output to your application program:

- Return and reason codes in registers from DFSMSrmm and the DFSMSrmm API.
- Return and reason codes from system services in structured fields.
- List header lines as formatted lines in structured fields.
- Messages as formatted lines or as message variables in structured fields.
- Report output data as formatted lines or as unformatted fields in structured fields.

The DFSMSrmm API does not return output data in the output buffer for every subcommand you issue using the API. See "Structured field introducers for output data for subcommands" on page 45 for information on each subcommand and the possible output data that the API returns as structured fields in your output buffer.

Description of structured fields

A structured field consists of:

- A structured field introducer (SFI)
- Data that follows the structured field introducer:

Part
Description

SFI
Structured field introducer. A structure with a minimum size of 8 bytes in this format:

Byte count
Description

2
Two-byte length. The length includes the length of the structured field introducer (8 bytes) and the length of the data following the structured field introducer.

3
Three-byte SFI identifier (ID)

1
One-byte SFI type modifier

1
One-byte (reserved)

1
One-byte data-type identifier

Data
Data following the structured field introducer, which can contain actual data, no data, binary zeros, or blank data.

See Appendix A, “Structured field introducers (SFIs),” on page 63 for descriptions of the structured field introducers that the DFSMSrmm API returns.

Structured fields can appear in any order. Write your application so it skips over any structured field it is not prepared to handle. This makes your application program less sensitive to changes like enhancements to DFSMSrmm that introduce new or different structured fields and sequences. You can update your application program when it is convenient to do so rather than being forced to do so because your application program no longer works.

In the examples that follow, <SFI>data denotes a structured field introducer (SFI) that is followed by data. In the examples, the term "SFI" is replaced with its descriptive name, for example: <data-set-name>. There is no association between the length of a particular structured field introducer and its descriptive name.

Requesting structured field introducer data format

You determine if the DFSMSrmm API returns line format or field format data to your application program. Line format contains fixed text and variable data that are formatted into lines. Line format is suitable for displaying at a terminal or for printing. Field format data consists only of structured field introducers and variable data.

You can request that the data be returned in line format when you specify the EDGXCI macro OUTPUT=LINES parameter. You can request that the data be returned in field format by specifying the OUTPUT=FIELDS parameter.

When you specify the EDGXCI macro OUTPUT=LINES parameter, the DFSMSrmm API returns the output lines in the same format as information returned by the DFSMSrmm RMM TSO subcommand.

In the examples that follow, assume that

```
A00001: RMMUSER.TSO.COMMAND1.
```

is only one data set on the volume

Requesting line format

Figure 17 on page 37 is an example of the line format data that the DFSMSrmm API returns when you specify the OUTPUT=LINES parameter. In the example, the request specifies the RMM TSO subcommand LISTDATASET RMMUSER.TAPE VOLUME(A00001). The request might produce the output that is shown in Figure 17 on page 37. The value for *<line>* is the SFI for each line and is followed by the data returned from specifying the RMM LISTDATASET subcommand.

```
<Begin DATASET Group>
  <line>Data set name = RMMUSER.TAPE
  <line>Volume        = A06061               Physical file sequence number = 1
  <line>Owner         = RMMUSER                        Data set sequence = 1
  <line>Create date   = 11/18/2016 Create time = 05:02:23 System ID         = EZU34
  <line>Expiration date = 11/21/2016                    Expir. time = 02:14:45
  <line>         set by = CATLG_DAYS        Original expir.date  =
  <line>LASTREF Extra Days = 0             WHILECATALOG          = OFF
  <line>Block size          = 3120         Block count           = 1
  <line>Data set size(KB)     = 97656
  <line>Physical size(KB)     = 0          Compression           = 0.00
  <line>Percent of volume     = 0          Total block count     = 1
  <line>Logical Record Length = 80         Record Format         = FB
  <line>Date last written     = 11/18/2016 Date last read        = 11/18/2016
  <line>Job name              = RMMUSERJ   Last job name         = RMMUSERJ
  <line>Step name             = WRITE      Last step name        = WRITE
  <line>Program name          = IEBGENER   Last program name     = IEBGENER
  <line>DD name               = SYSUT2     Last DD name          = SYSUT2
  <line>Device number         = 0590       Last Device number    = 0590
  <line>Management class      =            VRS management value  =
  <line>Storage group         =            VRS retention date    =
  <line>Storage class         =            VRS retained          = NO
  <line>Data class            =            Closed by Abend       = NO
  <line>                                   Deleted               = NO
  <line>VRSEL exclude         = YES        Catalog status        = UNKNOWN
  <line>Primary VRS details:
  <line>       Name           =
  <line>       Job name       =            Type                  =
  <line>       Subchain NAME  =            Subchain start date   =
  <line>Secondary VRS details:
  <line>       Value or class =
  <line>       Job name       =
  <line>       Subchain NAME  =            Subchain start date   =
  <line>Security Class   = UNCLASS    Description   = UNCLASSIFIED
  <line>BES key index    = 0
  <line>
  <line>Last Change information:
  <line>Date             = 11/18/2016  Time = 05:02:23  System  = EZU0000
  <line>User change date =             Time =           User ID = *OCE
  <line>
<End DATASET Group>
```

Figure 17. Example of list type of output using OUTPUT=LINES

Requesting field format

Figure 18 on page 38 is an example of the field format data that the DFSMSrmm API returns when you specify the OUTPUT=FIELDS parameter. Your request specifying LISTDATASET FIELD.TEST VOLUME(VOL001) subcommand might also produce the output shown in Figure 18 on page 38.

```
<Begin DATASET Group>
  <DSN  - Data Set Name           : 44, character        >
  <CJBN - Job Name                :  8, character        >
  <VOL  - Volume Serial           :  6, character        >
  <OWN  - Owner                   :  8, character        >
  <DSEQ - Data Set Sequence       :  4, bin(31)          >
  <TZ   - Time Zone               :  4, bin(31)          >
  <DEV  - Device Number           :  4, character        >
  <FILE - Physical File Sequence  :  4, bin(31)          >
  <CDTJ - Create Date             :  4, packed decimal   >
  <CTM  - Create Time             :  4, packed decimal   >
  <SYS  - Creating system ID      :  8, character        >
  <BLKS - Block Size              :  4, bin(31)          >
  <BLKC - Block Count             :  4, bin(31)          >
  <LRCL - Logical Record Length   :  4, bin(31)          >
  <RCFM - Record Format           :  4, character        >
  <DC   - Data Class              :  8, character        >
  <DLWJ - Date Last Written       :  4, packed decimal   >
  <DLRJ - Date Last Read/Referenced:  4, packed decimal   >
  <STEP - Step Name               :  8, character        >
  <DD   - DD Name                 :  8, character        >
  <MC   - Management Class        :  8, character        >
  <SG   - Storage Group Name      :  8, character        >
  <SC   - Storage Class           :  8, character        >
  <VMV  - VRS Management Value    :  8, character        >
  <RTDJ - Retention Date          :  4, packed decimal   >
  <VTYP - Primary VRS Type        :  1, bin(8)           >
  <VJBN - Primary VRS Job Name    :  8, character        >
  <VNME - Primary VRS Name        : 44, character        >
  <VSCN - Primary VRS Subchain name:  8, character       >
  <VSCD - Primary VRS Subchain date:  4, packed decimal  >
  <VRSR - VRS Retained            :  1, bin(8)           >
  <NME  - Security Class Name     :  8, character        >
  <CLS  - Security Class Descriptio: 32, character       >
  <ABND - Abend while open        :  1, bin(8)           >
  <CTLG - Catalog status          :  1, bin(8)           >
  <2JBN - Secondary VRS jobname mas:  8, character       >
  <2NME - Secondary VRS mask      :  8, character        >
  <2SCN - Secondary VRS subchain na:  8, character       >
  <2SCD - Secondary VRS subchain da:  4, packed decimal  >
  <BLKT - Total block count       :  4, bin(31)          >
  <CPGM - Creating program name   :  8, character        >
  <LPGM - Last used program name  :  8, character        >
  <LJOB - Last used job           :  8, character        >
  <LSTP - Last used step name     :  8, character        >
  <LDD  - Last used DD name       :  8, character        >
  <LDEV - Last Drive              :  4, character        >
  <DPCT - Percent of volume       :  1, bin(8)           >
  <XDTJ - Expiration Date         :  4, packed decimal   >
  <XDSB - Expiry date set by      :  1, bin(8)           >
  <OXDJ - Original Expiration Date :  4, packed decimal  >
  <DSS6 - Data Set Size           : 14, compound         >
  <LCDJ - Last Change Date        :  4, packed decimal   >
  <LCTM - Last Change Time        :  4, packed decimal   >
  <LCID - Last Change User ID     :  8, character        >
  <LCSI - Last Change System ID   :  8, character        >
  <LCUD - Last "User" Change Date  :  4, packed decimal  >
  <LCUT - Last "User" Change Time  :  4, packed decimal  >
  <DLTD - Deleted By Disposition Pr:  1, bin(8)          >
  <VEX  - VRSEL EXCLUDE on        :  1, bin(8)           >
  <BESK - CA Tape Encrytion key ind:  4, bin(31)         >
  <PSZ6 - Physical space used     : 14, compound         >
  <CRAT - Compression ratio in hund:  6, bin(31)         >
  <BLK6 - Total block count ( 64 bi:  8, bin(64)         >
  <LRED - LASTREF extra days      :  4, bin(31)          >
  <WCTL - WHILECATALOG for data set: 1, bin(8)           >
  <XTM - Expiration Time : 4, packed decimal             >
<End DATASET Group>
```

Figure 18. Example of output using OUTPUT=FIELDS

Figure 18 on page 38:

- Shows Begin and End group structured field introducers. In this example, <Begin DATASET Group> and <End DATASET Group>.
- Includes descriptive names used to identify structured field introducers. The SFI identifies the data type; and the long character <...> strings do not represent the actual size of the structured field introducers, which are only 8 bytes in length.
- Can appear to have no data. This is because structured fields can
  - Have no data (SFI only, as in this example), binary zeros, or blank characters.
  - Be omitted if they have no data.
- Shows that structured fields can be order independent. For example, VOL occurs before OWN for LISTDATASET (as shown in "LISTDATASET structured field introducers" on page 51) while OWN occurs before VOL for LISTPRODUCT (as shown in "LISTPRODUCT structured field introducers" on page 53).

- Shows that structured fields might not be in the same order as their corresponding positions in any line-format output.
- Shows variable-length fields.

Refer to Appendix D, “Hexadecimal example of an output buffer,” on page 153 for an example of an output buffer in hexadecimal representation.

Requesting types of output

The DFSMSrmm API can produce standard output and expanded output depending on the values you specify for the OUTPUT and EXPAND parameters as described in “EDGXCI parameters” on page 6.

The examples shown in “Requesting standard output” on page 39 and “Requesting expanded output” on page 39:

- Assume that there is only one data set on volume VOL001: OWNERONE.FIELD.TEST.
- Use SFI data type descriptions, such as DSN for data set name.
- Show maximum length values, without the term "bytes".
- Show the data type, such as character.

Requesting standard output

When you specify EXPAND=NO, your request specifying the SEARCHDATASET VOLUME(VOL001) subcommand might produce the output that is shown in Figure 19 on page 39.

```
<Begin DATASET Group>
  <DSN  - Data Set Name            : 44, character         >RMMUSER.DATA01
  <VOL  - Volume Serial            :  6, character         >V10000
  <OWN  - Owner                    :  8, character         >RMMUSER
  <TZ   - Time Zone                :  4, bin(32)           >x'FFFF9D90'
  <CDTJ - Create Date              :  4, packed decimal    >x'2007339F'
  <CTM  - Create Time              :  4, packed decimal    >x'0116362F'
  <FILE - Physical File Sequence   :  4, bin(32)           >x'00000001'
  <RTDJ - Retention Date           :  4, packed decimal    >x'2010345F'
  <XDTJ - Expiration Date          :  4, packed decimal    >x'2010344F'
<End DATASET Group>
```

Figure 19. Example of search type of output using EXPAND=NO

Refer to Appendix D, “Hexadecimal example of an output buffer,” on page 153 for a hexadecimal representation and discussion of the contents of the output buffer shown in Figure 19 on page 39.

Requesting expanded output

The DFSMSrmm API can provide expanded output for the DFSMSrmm TSO RMM SEARCHDATASET, SEARCHPRODUCT, SEARCHVOLUME, and SEARCHVRS subcommands when you specify OUTPUT=FIELDS and EXPAND=YES or use the default EXPAND=YES in your application program.

The DFSMSrmm API does not provide expanded data for the DFSMSrmm TSO RMM SEARCHBIN or SEARCHRACK subcommands.

When you specify OUTPUT=FIELDS and EXPAND=YES, your SEARCHDATASET VOLUME(VOL001) subcommand might produce the output that is shown in Figure 20 on page 40.

```
<Begin DATASET Group>
  <DSN  - Data Set Name            : 44, character        >RMMUSER.TAPE
  <CJBN - Job Name                 :  8, character        >RMMUSERJ
  <VOL  - Volume Serial            :  6, character        >A06061
  <OWN  - Owner                    :  8, character        >RMMUSER
  <DSEQ - Data Set Sequence        :  4, bin(31)          >x'00000001'
  <TZ   - Time Zone                :  4, bin(31)          >x'FFFF9D90'
  <DEV  - Device Number            :  4, character        >0590
  <FILE - Physical File Sequence   :  4, bin(31)          >x'00000001'
  <CDTJ - Create Date              :  4, packed decimal   >x'2007339F'
  <CTM  - Create Time              :  4, packed decimal   >x'0116381F'
  <SYS  - Creating system ID       :  8, character        >EZU0000
  <BLKS - Block Size               :  4, bin(31)          >x'00000C30'
  <BLKC - Block Count              :  4, bin(31)          >x'00000001'
  <LRCL - Logical Record Length    :  4, bin(31)          >x'00000050'
  <RCFM - Record Format            :  4, character        >FB
  <DC   - Data Class               :  8, character        >
  <DLWJ - Date Last Written        :  4, packed decimal   >x'2007339F'
  <DLRJ - Date Last Read/Referenced:  4, packed decimal   >x'2007339F'
  <STEP - Step Name                :  8, character        >WRITE
  <DD   - DD Name                  :  8, character        >SYSUT2
  <MC   - Management Class         :  8, character        >
  <SG   - Storage Group Name       :  8, character        >
  <SC   - Storage Class            :  8, character        >
  <VMV  - VRS Management Value     :  8, character        >
  <RTDJ - Retention Date           :  4, packed decimal   >
  <VTYP - Primary VRS Type         :  1, bin(8)           >x'00'
  <VJBN - Primary VRS Job Name     :  8, character        >
  <VNME - Primary VRS Name         : 44, character        >
  <VSCN - Primary VRS Subchain name:  8, character        >
  <VSCD - Primary VRS Subchain date:  4, packed decimal   >
  <VRSR - VRS Retained             :  1, bin(8)           >x'00'
  <NME  - Security Class Name      :  8, character        >
  <CLS  - Security Class Descriptio: 32, character        >
  <ABND - Abend while open         :  1, bin(8)           >x'00'
  <CTLG - Catalog status           :  1, bin(8)           >x'00'
  <2JBN - Secondary VRS jobname mas:  8, character        >
  <2NME - Secondary VRS mask       :  8, character        >
  <2SCN - Secondary VRS subchain na:  8, character        >
  <2SCD - Secondary VRS subchain da:  4, packed decimal   >
  <BLKT - Total block count        :  4, bin(31)          >x'00000001'
  <CPGM - Creating program name    :  8, character        >IEBGENER
  <LPGM - Last used program name   :  8, character        >IEBGENER
  <LJOB - Last used job            :  8, character        >RMMUSERJ
  <LSTP - Last used step name      :  8, character        >WRITE
  <LDD  - Last used DD name        :  8, character        >SYSUT2
  <LDEV - Last Drive               :  4, character        >0590
  <DPCT - Percent of volume        :  1, bin(8)           >x'00'
  <XDTJ - Expiration Date          :  4, packed decimal   >x'2007344F'
  <XDSB - Expiry date set by       :  1, bin(8)           >x'06'
  <OXDJ - Original Expiration Date :  4, packed decimal   >
  <DSS6 - Data Set Size            : 14, compound         >x'010303010A060000000000017D78'
  <LCDJ - Last Change Date         :  4, packed decimal   >x'2011322F'
  <LCTM - Last Change Time         :  4, packed decimal   >x'0502236F'
  <LCID - Last Change User ID      :  8, character        >*OCE
  <LCSI - Last Change System ID    :  8, character        >EZU0000
  <LCUD - Last "User" Change Date  :  4, packed decimal   >
  <LCUT - Last "User" Change Time  :  4, packed decimal   >
  <DLTD - Deleted By Disposition Pr:  1, bin(8)           >x'00'
  <VEX  - VRSEL EXCLUDE on         :  1, bin(8)           >x'01'
  <BESK - CA Tape Encrytion key ind:  4, bin(31)          >x'00000000'
  <PSZ6 - Physical space used      : 14, compound         >x'010303010A060000000000000000'
  <CRAT - Compression ratio in hund:  6, bin(31)          >x'00000000'
  <BLK6 - Total block count ( 64 bi:  8, bin(64)          >x'0000000000000001'
  <LRED - LASTREF extra days       :  4, bin(31)          >x'00000000'
  <WCTL - WHILECATALOG for data set:  1, bin(8)           >x'00'
  <XTM - Expiration time           : 4, packed decimal    > x'0127118F'
  <CTRT - Catalog retained status  : 1, bin(8)            > x'00'
  <FRXP - FORCEEXPIRE status       : 1, bin(8)            > x'00'
<End DATASET Group>
```

Figure 20. Example of search type of output using OUTPUT=FIELDS, EXPAND=YES

Accessing return and reason codes

DFSMSrmm returns return codes and reason codes to your application program in the general purpose registers and also as data in your output buffer as follows:

- Return codes and reason codes issued as a result of processing of your subcommand request. Refer to *z/OS DFSMSrmm Managing and Using Removable Media* for information about these codes.
- Return codes and reason codes associated with the API itself. These are the return codes and reason codes listed in "EDGXCI return and reason codes" on page 9 for macro EDGXCI.
- Return and reason codes from system services. DFSMSrmm uses various system services, such as catalog services, to process the subcommands from your application program. When DFSMSrmm receives a non-zero return code from a system service, the DFSMSrmm API places the return code and associated reason code in your output buffer as structured fields, along with a name to identify the service. See "System return and reason code structured field introducers" on page 43 for more information.

Accessing messages and message variables

The DFSMSrmm API can return messages and message variables in your output buffer. Figure 21 on page 41 show how messages are returned in line format when you specify the OUTPUT=LINES parameter and field format when you specify the OUTPUT=FIELDS parameter.

```
  <message line>message text
  <message line>message text

or

  <Begin MESSAGE group>
    <message number  >number
    <message variable>variable
  <End MESSAGE group>
  <Begin MESSAGE group>
    <message number  >number
    <message variable>variable
  <End MESSAGE group>
```

Figure 21. Message and message variable structured fields

Refer to "Messages and message variables structured field introducers" on page 43 for information about which messages can be placed in your output buffer.

Interpreting date format and time format

DFSMSrmm dates are in packed decimal format: yyyydddC, where yyyyddd is a Julian date and C is a standard packed-decimal sign character. The date formats used are returned in internal format and can be interpreted as follows:

- Interpret 9999366 as PERMANENT retention date format.
- Interpret 9999365 as PERMANENT retention date format.
- Interpret 9800000 as WHILECATLG retention date format.
- Interpret 98ccccc as CYCL/ccccc retention date format.
- Interpret 0000098 as CATRETPD retention date format.
- Interpret 0000099 as KeptByCatlg retention date format. (If the data set is cataloged and WHILECATALOG(ON) is set, SEARCHDATASET SFI X'8C6000' contains the special KeptByCatlg qualifier X'0000099F', while LISTDATASET SFI X'8C6000' always contains the expiration date.)
- Interpret yyyyddd as yyyy/mm/dd, yyyy/dd/mm, mm/dd/yyyy, dd/mm/yyyy, dd/yyyy/mm, mm/yyyy/dd.

DFSMSrmm also returns time in packed decimal format: hhmmsstC, where hhmmsst is the time in hours, minutes, seconds, and tenths of seconds and C is a standard packed-decimal sign character.

Using different time zones

Default dates and times are returned in the time zone of the DFSMSrmm system processing the subcommand. The TZ SFI provides the time zone offset so if necessary, the application can convert

dates and times to any other required time zone. When issuing subcommands that specify date or time values, such as ADDDATASET or CHANGEVOLUME, you can specify the TZ operand to indicate to the DFSMSrmm system the time zone offset the application is using. DFSMSrmm converts dates and times to UTC/GMT/local time in order to store them in the DFSMSrmm control data set. Refer to *z/OS DFSMSrmm Implementation and Customization Guide* for more information on creating or updating the DFSMSrmm control data set control record and setting up DFSMSrmm common time support.

Identifying structured field introducers

A structured field introducer (SFI) is a structure that identifies one line or field of output data from another. The DFSMSrmm API returns these types of structured field introducers in your output buffer:

- Structured field introducers that begin and end a resource group as described in "Begin and End Resource groups" on page 42.
- Structured field introducers that introduce a single line of output data, as described in:
  - "System return and reason code structured field introducers" on page 43
  - "Messages and message variables structured field introducers" on page 43
  - "ADD-Type of subcommands" on page 45
  - "CHANGE-Type of subcommands" on page 46
  - "DELETE-Type of subcommands" on page 46
  - "GETVOLUME subcommand" on page 46
  - "LIST-Type of subcommands" on page 47
  - "SEARCH-Type of subcommands" on page 55

This notation indicates an SFI:

```
<xxxx - descriptive name       : data length, data type :  >
```

where "xxxx" is a character type of mnemonic. In your application program, you need to use the 3-byte or 4-byte hexadecimal identifiers for structured field introducers.

Appendix A, "Structured field introducers (SFIs)," on page 63 describes all the structured fields that the DFSMSrmm API can return to your application program.

Appendix B, "Structured field introducers by subcommand," on page 87 shows all of the structured field introducers by subcommand.

The DFSMSrmm API does not return information for all subcommands. For example, the DFSMSrmm API does not produce structured fields for a successful ADDBIN subcommand request.

Begin and End Resource groups

In the previous examples, you saw that output structured fields were grouped by a pair of unique structured field introducers as shown in Figure 22 on page 42.

```
<Begin DATASET group>
  <..          >data set name
  <..          >volume id
<End DATASET group>
```

Figure 22. Begin and End Resource group SFI sequence

The Begin and End Resource group structured field introducers identify when output for a particular resource, such as a data set, begins and ends. The pairs of Begin and End Resource group structured field introducers are shown in Figure 23 on page 43.

```
  <Begin BIN group>       <End BIN group>
  <Begin CONTROL group>   <End CONTROL group>
  <Begin DATASET group>   <End DATASET group>
  <Begin MESSAGE group>   <End MESSAGE group>
  <Begin OWNER group>     <End OWNER group>
  <Begin PRODUCT group>   <End PRODUCT group>
  <Begin RACK group>      <End RACK group>
  <Begin VOLUME group>    <End VOLUME group>
  <Begin VRS group>       <End VRS group>
```

Figure 23. Begin and End Resource group SFI pairs

In addition to identifying the beginning and ending of output for a particular resource, the Begin and End Resource group structured field introducers shown in Figure 24 on page 43 are used to differentiate one subgroup of data from another in the output the DFSMSrmm API returns for the LISTCONTROL, LISTVOLUME, SEARCHVOLUME, LISTPRODUCT, and SEARCHPRODUCT subcommands.

```
  <Begin ACCESS group>   <End ACCESS group>
  <Begin ACTIONS group>  <End ACTIONS group>
  <Begin CNTL group>     <End CNTL group>
  <Begin DEFAULT group>  <End DEFAULT group>
  <Begin LOCDEF group>   <End LOCDEF group>
  <Begin MEDINF group>   <End MEDINF group>
  <Begin MNTMSG group>   <End MNTMSG group>
  <Begin MOVES group>    <End MOVES group>
  <Begin OPENRULE group> <End OPENRULE group>
  <Begin OPTION group>   <End OPTION group>
  <Begin PRTITION group> <End PRTITION group>
  <Begin PRODVOL group>  <End PRODVOL group>
  <Begin REJECT group>   <End REJECT group>
  <Begin SECCLS group>   <End SECCLS group>
  <Begin STAT group>     <End STAT group>
  <Begin STATUS group>   <End STATUS group>
  <Begin STORE group>    <End STORE group>
  <Begin TASKS group>    <End TASKS group>
  <Begin VLPOOL group>   <End VLPOOL group>
  <Begin VOL group>      <End VOL group>
```

Figure 24. Begin and End Resource group SFI pairs for subgroups

Groups and subgroups, such as MESSAGE and SECCLS, are repeated as often as necessary to differentiate resources.

System return and reason code structured field introducers

When DFSMSrmm receives a non-zero return code from a system service, the system return code and associated reason code are put into your output buffer as shown in Figure 25 on page 43. DFSMSrmm issues return code 116 and reason code 06 when an error like this occurs.

```
<Begin SYSRETC group>
  <SVCN - service name            : 16 , character:        >
  <RTNC - return code             : 4  , bin(32):          >
  <RSNC - reason code             : 4  , bin(32):          >
<End SYSRETC group>
```

Figure 25. System return and reason codes

The DFSMSrmm API returns the same structured field introducers for both line format and field format.

Messages and message variables structured field introducers

When messages or message variables are returned to you as output data, they are put into your output buffer as structured fields as shown in Figure 26 on page 44.

```
    <MSGL - message line            : nn , character:         >
    <MSGL - message line            : nn , character:         >

or

  <Begin MESSAGE group>
    <MSGN - message number          : 8  , character:         >
    <xxx  - variable>
  <End MESSAGE group>
  <Begin MESSAGE group>
    <MSGN - message number          : 8  , character:         >
    <xxx  - variable>
  <End MESSAGE group>
```

Figure 26. Structured field introducers for messages and message variables

When you use the CONTINUE operand on any SEARCH subcommand, the DFSMSrmm API returns the continue information at the message group with the CONT SFI as shown in Figure 27 on page 44.

```
<Begin VOLUME group>
  <Begin MESSAGE group>
    <MSGN - message number           : 8 , character:         >
    <ENTN - number of entries        : 4 , bin(1):            >
  <End MESSAGE group>
  <Begin MESSAGE group>
    <MSGN - message number           : 8 , character:         >
    <CONT -continue information      :84 , character:         >
  <End MESSAGE group>
<End VOLUME group>
```

Figure 27. Message group with the CONT SFI

When you specify OUTPUT=LINES, messages issued by DFSMSrmm are placed in your output buffer using the LINE SFI.

When you specify OUTPUT=FIELDS, only the messages listed in Table 12 on page 44 are placed in your output buffer. These messages, some of which are issued only in conjunction with a subcommand parameter such as POOL or COUNT, are included in the output because they contain data and codes that can be especially useful to your application. Your application program should use the return and reason codes that it receives rather than messages to determine whether or not the subcommand request was successful.

Table 12 on page 44 lists:

- The structured field introducers that follow the <MSGN> SFI
- The applicable subcommands
- A non-inclusive list of the return codes (RC) and reason codes (RSN).

Table 12. Message related structured field introducers

Message	SFI ID(s)	Subcommand(s)	RC	RSN(s)
EDG3010	ENTN	All SEARCH subcommands when no (0) entry is returned	4	8
EDG3011	ENTN	All SEARCH subcommands when 1 entry returned	0 4	0 2 and 4
EDG3012	ENTN	All SEARCH subcommands when > 1 entry returned	0 4	0 2 and 4
EDG3013	VOL	AV	12	many
EDG3014	CNT	AV	12	many
EDG3015	OWN VOL	GV	0	0
EDG3016	RCK	AV CV	0	0
EDG3017	RCK	AB AR	12	18 68 70

Table 12. Message related structured field introducers (continued)				
Message	**SFI ID(s)**	**Subcommand(s)**	**RC**	**RSN(s)**
EDG3018	CNT	AB AR	12	18 68 70
EDG3019	RCK	DB DR	12	many
EDG3020	CNT	DB DR	12	many
EDG3025	CONT	All SEARCH subcommands	4	2
EDG3277	FRC FRS	AV CV	12	122
EDG3278	CSG	AV CV	12	124
EDG3288	FRC FRS VOL	CV DV	12	132
EDG3289	FRC FRS	CV	12	134
EDG3292	CLIB	AV CV	12	140
EDG3301	FRC FRS	AV CV GV	12	152
EDG3310	CLIB	CV DV	12	170
EDG3311	FRC FRS	AV CV DV	12	172
EDG3314	MEDN	CV	12	176
EDG3328	KEYF KEYT TYPF TYPT	SD SV	4	12

For a detailed explanation of these messages, see *z/OS MVS System Messages, Vol 5 (EDG-GLZ)*. For DFSMSrmm return and reason codes, see *z/OS DFSMSrmm Managing and Using Removable Media*.

Structured field introducers for output data for subcommands

When you specify OUTPUT=LINES, the DFSMSrmm API returns output data, except for system return and reason codes, as formatted lines in structured fields. The structured fields are introduced by the <LINE> and <MSGL> structured field introducers as shown in Figure 28 on page 45. DFSMSrmm places system return codes and reason codes in your output buffer as described in "System return and reason code structured field introducers" on page 43.

```
<Begin resource group>
  <LINE - Formatted output line   : nn , character:      >
  <LINE - Formatted output line   : nn , character:      >
  <MSGL - Formatted output message: nn , character:      >
  <MSGL - Formatted output message: nn , character:      >
<End resource group>
```

Figure 28. Formatted lines

When you specify OUTPUT=FIELDS, the DFSMSrmm API returns output data as unformatted data in structured fields.

ADD-Type of subcommands

The DFSMSrmm ADD-type of subcommands are: ADDBIN, ADDDATASET, ADDOWNER, ADDPRODUCT, ADDRACK, ADDVOLUME, and ADDVRS. You use these subcommands to add information to the DFSMSrmm control data set.

The DFSMSrmm API returns information under these conditions:

- You specify the ADDVOLUME subcommand with the POOL operand. The DFSMSrmm API returns the rack number that is assigned to the volume in the format as shown in Figure 29 on page 46.
- An error occurs for specific return and reason code combinations described in "Messages and message variables structured field introducers" on page 43 and "Structured field introducers for return and reason codes" on page 65.

```
<Begin VOLUME group>
  <Begin MESSAGE group>
    <MSGN - message number           : 8  , character:         >
    <RCK  - rack or bin number       : 6  , character:         >
  <End MESSAGE group>
<End VOLUME group>
```

Figure 29. Structured field introducers for ADDVOLUME with OUTPUT=FIELDS

CHANGE-Type of subcommands

The DFSMSrmm CHANGE-type of subcommands are: CHANGEDATASET, CHANGEOWNER, CHANGEPRODUCT,and CHANGEVOLUME. You use these subcommands to change information in the DFSMSrmm control data set.

The DFSMSrmm API returns information when:

- You specify the CHANGEVOLUME subcommand with the POOL operand. The DFSMSrmm API returns the rack number that is assigned to the volume in the format as shown in Figure 30 on page 46.
- When an error occurs for specific return and reason code combinations described in "Messages and message variables structured field introducers" on page 43 and "Structured field introducers for return and reason codes" on page 65.

```
<Begin VOLUME group>
  <Begin MESSAGE group>
    <MSGN - message number           : 8  , character:         >
    <RCK  - rack or bin number       : 6  , character:         >
  <End MESSAGE group>
<End VOLUME group>
```

Figure 30. SFIs for CHANGEVOLUME with OUTPUT=FIELDS

DELETE-Type of subcommands

The DFSMSrmm DELETE-type of subcommands are: DELETEBIN, DELETEDATASET, DELETEOWNER, DELETEPRODUCT, DELETERACK, DELETEVOLUME, and DELETEVRS. You use these subcommands to delete information from the DFSMSrmm control data set.

The DFSMSrmm API returns information when an error occurs for specific return and reason code combinations described in "Messages and message variables structured field introducers" on page 43 and "Structured field introducers for return and reason codes" on page 65.

GETVOLUME subcommand

You use the RMM GETVOLUME subcommand to obtain a volume from DFSMSrmm.

The DFSMSrmm API returns information when:

- The GETVOLUME request was successful. The DFSMSrmm API returns volume information and owner information as shown in Figure 31 on page 47.
- When an error occurs, and then only for specific return and reason code combinations described in "Messages and message variables structured field introducers" on page 43 and "Structured field introducers for return and reason codes" on page 65.

```
<Begin VOLUME group>
  <Begin MESSAGE group>
    <MSGN - message number          : 8  , character:         >
    <VOL  - volume serial           : 6  , character:         >
    <OWN  - owner                   : 8  , character:         >
  <End MESSAGE group>
<End VOLUME group>
```

Figure 31. Structured field introducers for GETVOLUME with OUTPUT=FIELDS

LIST-Type of subcommands

The DFSMSrmm LIST-type of subcommands are: LISTBIN, LISTCONTROL, LISTDATASET, LISTOWNER, LISTPRODUCT, LISTRACK, LISTVOLUME, and LISTVRS. You use these subcommands to obtain information from the DFSMSrmm control data set about a single resource.

The DFSMSrmm API returns output data for LIST type of subcommands as structured fields when you specify OUTPUT=FIELDS. The structured field introducers for each type of LIST subcommand are found in:

- "LISTBIN structured field introducers" on page 47
- "LISTCONTROL structured field introducers" on page 47
- "LISTDATASET structured field introducers" on page 51
- "LISTOWNER structured field introducers" on page 52
- "LISTPRODUCT structured field introducers" on page 53
- "LISTRACK structured field introducers" on page 53
- "LISTVOLUME structured field introducers" on page 53
- "LISTVRS structured field introducers" on page 55

LISTBIN structured field introducers

The structured field introducers produced for the LISTBIN subcommand with OUTPUT=FIELDS are:

```
<Begin RACK/BIN Group>
  <RCK  - Rack or Bin Number       :  6, character         >
  <VOL  - Volume Serial            :  6, character         >
  <RST  - Rack or Bin Status       :  1, bin(8)            >
  <LOC  - Location                 :  8, character         >
  <MEDN - Media Name               :  8, character         >
  <MIV  - Moving-In Volume         :  6, character         >
  <MOV  - Moving-Out Volume        :  6, character         >
  <OVOL - Old Volume               :  6, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
<End RACK/BIN Group>
```

LISTCONTROL structured field introducers

The structured field introducers produced for the LISTCONTROL subcommand with OUTPUT=FIELDS differ depending on whether the STATUS operand is specified.

The structured field introducers produced for the LISTCONTROL DEFTABLE subcommand with OUTPUT=FIELDS are:

```
<Begin CONTROL Group>
  <Begin CNTL Group>
    <TZ   - Time Zone               :  4, bin(31)           >
    <MTP  - CDS type                :  1, bin(8)            >
    <MDTJ - CDS Create Date         :  4, packed decimal    >
    <MTM  - CDS Create Time         :  4, packed decimal    >
```

```
    <UDTJ - CDS Last update date     :  4, packed decimal    >
    <UTM  - CDS Last update time     :  4, packed decimal    >
    <JRNU - Journal Percentage Used  :  2, bin(15)           >
    <JRNF - JOURNALFULL Parmlib Value:  2, bin(15)           >
    <JRNS - Journal status           :  1, bin(8)            >
    <BDTJ - Last CDS Backup Date     :  4, packed decimal    >
    <BTM  - Last CDS Backup Time     :  4, packed decimal    >
    <JBDT - Last journal backup date :  4, packed decimal    >
    <JBTM - Last journal backup time :  4, packed decimal    >
    <XDTJ - Expiration Date          :  4, packed decimal    >
    <XTM  - Last Inven Mgt Expir Time:  4, packed decimal    >
    <RDTJ - Last CDS Extract Date    :  4, packed decimal    >
    <RTM  - Last CDS Extract Time    :  4, packed decimal    >
    <DDTJ - Delete/Store Date        :  4, packed decimal    >
    <DTM  - Last Store update run tim:  4, packed decimal    >
    <SOSJ - Last XPROC Start Date    :  4, packed decimal    >
    <SOST - Last XPROC Start Time    :  4, packed decimal    >
    <VDTJ - Last Inven Mgt Proc Date :  4, packed decimal    >
    <VTM  - Last Inven Mgt VRS Time  :  4, packed decimal    >
    <LRK  - # Library Rack Numbers   :  4, bin(31)           >
    <FRK  - Free Rack Num in Library :  4, bin(31)           >
    <LBN  - Bin Numbers in LOCAL     :  4, bin(31)           >
    <FLB  - Free Bin Numbers in LOCAL:  4, bin(31)           >
    <DBN  - Bin Numbers in DISTANT   :  4, bin(31)           >
    <FDB  - Free Bins in DISTANT Loc :  4, bin(31)           >
    <RBN  - # Bin Numbers in REMOTE  :  4, bin(31)           >
    <FRB  - Free Bin Numbers in REMOT:  4, bin(31)           >
    <CACT - Control Active Functions :  1, bit(8)            >
    <CSDT - Catalog Synchronize date :  4, packed decimal    >
    <CSTM - Catalog Synchronize time :  4, packed decimal    >
    <FCSP - Catalog Synch in progress:  1, bin(8)            >
    <CSVE - Stacked volume enabled   :  1, bin(8)            >
    <X100 - EDGUX100 exit status     :  1, bin(8)            >
    <X200 - EDGUX200 exit status     :  1, bin(8)            >
    <X300 - EDGUX300 exit status     :  1, bin(8)            >
    <EBIN - Extended Bin Status      :  1, bin(8)            >
    <CDSU - CDS percentage used      :  2, bin(15)           >
    <CSHN - Client/Server host name  : 63, character         >
    <CSIP - Client/Server IP address : 15, character         >
    <UTC  - Common time status enable:  1, bin(8)            >
    <CDSQ - CDS id ENQ name enabled  :  1, bin(8)            >
    <CDSF - CDSFULL Parmlib Value    :  2, bin(15)           >
    <RMID - RMM started procedure nam: 17, character         >

  <End CNTL Group>
  <Begin OPTION Group>
    <OPM  - Operating Mode           :  1, bin(8)            >
    <DRP  - Default Retention Period :  4, bin(31)           >
    <MRP  - Maximum Retention Period :  4, bin(31)           >
    <CRP  - CATRETPD Retention Period:  4, bin(31)           >
    <MDS  - CDS Data Set Name        : 44, character         >
    <JDS  - Journal Name             : 44, character         >
    <JRNF - JOURNALFULL Parmlib Value:  2, bin(15)           >
    <CATS - CATSYSID value           :  1, bin(8)            >
    <SOSP - Scratch Procedure Name   :  8, character         >
    <BKPP - Backup Procedure Name    :  8, character         >
    <IPL  - Data Check Required in IP:  1, bin(8)            >
    <DTE  - Installation Date Format :  1, bin(8)            >
    <RCF  - Installation RACF Support:  1, bin(8)            >
    <AUD  - SMF Audit Record Number  :  2, bin(15)           >
    <SSM  - SMF Security Record Numbe:  2, bin(15)           >
    <CDS  - Control Data Set ID      :  8, character         >
    <SLM  - MAXHOLD Value            :  2, bin(15)           >
    <LCT  - Default Lines per Page   :  2, bin(15)           >
    <SID  - SMF System ID            :  8, character         >
    <BLP  - BLP Option               :  1, bin(8)            >
    <NOT  - Notify                   :  1, bin(8)            >
    <UNC  - Uncatalog Option         :  1, bin(8)            >
    <VRJ  - VRS Job Name             :  1, bin(8)            >
    <MSGF - Case of Message Text     :  1, bin(8)            >
    <MOP  - Master Overwrite         :  1, bin(8)            >
    <ACCT - Accounting Source        :  1, bin(8)            >
    <VCHG - VRSCHANGE Value          :  1, bin(8)            >
    <VRSL - VRSEL Value              :  1, bin(8)            >
    <PSFX - Parmlib Member Suffix    :  2, character         >
    <PSF2 - Parmlib Member Suffix 2  :  2, character         >
    <VACT - VRSMIN action            :  1, bin(8)            >
    <VMIN - VRSMIN Count Value       :  4, bin(31)           >
    <JRNT - Journal transaction      :  1, bin(8)            >
    <VDRA - VRS Drop Action          :  1, bin(8)            >
    <VDRC - VRS Drop Count           :  4, bin(31)           >
    <VDRP - VRS Drop Percentage      :  2, bin(15)           >
```

```
  <VREA - VRS Retain Action        :  1, bin(8)           >
  <VREC - VRS Retain Count         :  4, bin(31)          >
  <VREP - VRS Retain Percentage    :  2, bin(15)          >
  <XDRA - EXPDT Drop Action        :  1, bin(8)           >
  <XDRC - EXPDT Drop Count         :  4, bin(31)          >
  <XDRP - EXPDT Drop Percentage    :  2, bin(15)          >
  <DSPD - Disposition DD name      :  8, character        >
  <DSPM - Disposition message prefi:  8, character        >
  <RTBY - Retain by                :  1, bin(8)           >
  <MVBY - Move by                  :  1, bin(8)           >
  <GDGC - GDG cycleby              :  1, bin(8)           >
  <GDGD - GDG duplicate            :  1, bin(8)           >
  <PDA  - PDA state                :  1, bin(8)           >
  <PDAC - PDA block count          :  1, bin(8)           >
  <PDAS - PDA block size           :  1, bin(8)           >
  <PDAL - PDA log state            :  1, bin(8)           >
  <TVXP - Extradays retention      :  1, bin(8)           >
  <TVXD - Extradays for TVEXTPURGE :  4, bin(31)          >
  <SMP  - System managed tape purge:  1, bin(8)           >
  <SMU  - System managed tape updat:  1, bit(8)           >
  <ACS  - SMS ACS support          :  1, bin(8)           >
  <PACS - Pre-ACS support          :  1, bin(8)           >
  <RUB  - Reuse Bin at             :  1, bin(8)           >
  <CMDD - Command Auth DSN         :  1, bin(8)           >
  <CMDO - Command Auth Owner       :  1, bin(8)           >
  <MEDN - Media Name               :  8, character        >
  <LCTK - Local Task               :  4, bin(31)          >
  <SSTY - Subsystem type           :  1, bin(8)           >
  <SRHN - Server host name         : 63, character        >
  <SRIP - Server IP Address        : 15, character        >
  <SRPN - Server port number       :  4, bin(31)          >
  <SRTK - Server task              :  4, bin(31)          >
  <RM   - Retention Method         :  1, bin(8)           >
  <LRED - LASTREF extra days       :  4, bin(31)          >
  <EXRB - EXPDT RetainBy           :  1, bin(8)           >
  <MCAT - Management class attribut:  1, bin(8)           >
  <DEFC - DEFTABLE Entries count   :  2, bin(15)          >
  <GDRP - Default RETPD, RM=EXPDT G:  4, bin(31)          >
  <NDRP - Default RETPD, RM=EXPDT n:  4, bin(31)          >
  <GWCT - Default WHILECATALOG, RM=:  1, bin(8)           >
  <NWCT - Default WHILECATALOG, RM=:  1, bin(8)           >
  <CTLD - Catalog Days             :  4, bin(31)          >
  <DSFX - Active DEFAULTS table suffix:  2, character     >
  <EDM - Managed externally        :  1, bin(8)           >
  <DSXP - DSNEXPIRE                :  1, bin(8)           >
  <HSKP - HouseKeeping             :  1, bin(8)           >


<End OPTION Group>
<Begin SECCLS Group>
  <SEC  - Security Class Number    :  1, bin(8)           >
  <NME  - Security Class Name      :  8, character        >
  <SCST - Security Class Status    :  1, bit(8)           >
  <CLS  - Security Class Descriptio: 32, character        >
<End SECCLS Group>
<Begin VLPOOL Group>
  <PID  - Pool Prefix              :  6, character        >
  <PSN  - Pool Definition System ID:  8, character        >
  <PRF  - Pool Def RACF Option     :  1, bin(8)           >
  <PTP  - Pool Def Pool Type       :  1, bin(8)           >
  <XDC  - Expiration Date Check    :  1, bin(8)           >
  <ACT  - Action on Release        :  1, bit(8)           >
  <SCRM - Scratch mode             :  1, bin(8)           >
  <PLN  - Pool Name                :  8, character        >
  <MEDN - Media Name               :  8, character        >
  <PDS  - Pool Description         : 40, character        >
  <MOP  - Master Overwrite         :  1, bin(8)           >
<End VLPOOL Group>
<Begin MNTMSG Group>
  <MID  - Mount Message ID         : 12, character        >
  <SMI  - Offset to Message ID     :  2, bin(15)          >
  <OVL  - Offset to Volume Serial  :  2, bin(15)          >
  <OPL  - Offset Rack Num or Pool I:  2, bin(15)          >
<End MNTMSG Group>
<Begin REJECT Group>
  <GRK  - Generic Rack Number      :  6, character        >
  <TAC  - Reject Type              :  1, bin(8)           >
<End REJECT Group>
<Begin LOCDEF Group>
  <LDDF - Location Definition Exist:  1, bin(8)           >
  <LDLC - Location Name            :  8, character        >
```

```
    <LDMT - Location Management Type :  1, bin(8)           >
    <LDLT - Location Type            :  1, bin(8)           >

    <LDPR - Location Priority        :  4, bin(31)          >
    <LDAM - Location Automove        :  1, bin(8)           >
    <LDMN - Location Media Name      :  8, character        >
  <End LOCDEF Group>
  <Begin MEDINF Group>
    <MDNF - Media Information Name   :  8, character        >
    <MEDT - Media Type               :  1, bin(8)           >
    <MDTX - External Media Type      :  8, character        >
    <MEDR - Media Recording Format   :  1, bin(8)           >
    <MDRX - External Recording Techn.:  8, character        >
    <VCAP - Volume capacity          :  4, bin(31)          >
    <MDRP - MEDINF Replace Policy Per:  4, bin(31)          >
    <MDRT - MEDINF Replace Policy Tem:  4, bin(31)          >
    <MDRW - MEDINF Replace Policy Wri:  4, bin(31)          >
    <MDRA - MEDINF Replace Policy Age:  4, bin(31)          >
  <End MEDINF Group>
  <Begin PRTITION Group>
    <PTVL - Volume Serial Number     :  6, character        >
    <PTVS - Volume Range Start       :  6, character        >
    <PTVE - Volume Range End         :  6, character        >
    <PTTP - Type of Partition Entry  :  1, bin(8)           >
    <PTSA - SMT Action for Partition :  1, bin(8)           >
    <PTNA - NOSMT Action for Partitio:  1, bin(8)           >
    <PTNL - Location Name            :  8, character        >
  <End PRTITION Group>
  <Begin OPENRULE Group>
    <ORVL - Volume Serial Number     :  6, character        >
    <ORVS - Volume Range Start       :  6, character        >
    <ORVE - Volume Range End         :  6, character        >
    <ORTP - Type of Openrule Entry   :  1, bin(8)           >
    <ORIA - Input Action             :  1, bin(8)           >
    <ORII - Input Ignore Condition   :  1, bit(8)           >
    <ORIR - Input Reject Condition   :  1, bit(8)           >

    <OROA - Output Action            :  1, bin(8)           >
    <OROI - Output Ignore Condition  :  1, bit(8)           >

    <OROR - Output Reject Condition  :  1, bit(8)           >
  <End OPENRULE Group>
  <Begin ACTIONS Group>
    <ACT  - Action on Release        :  1, bit(8)           >
    <AST  - Action Status            :  1, bit(8)           >
  <End ACTIONS Group>
  <Begin MOVES Group>
    <MFR  - Source Location Name     :  8, character        >
    <MST  - Move Status              :  1, bin(8)           >
    <MTO  - Target Location Name     :  8, character        >
    <MTY  - Move Type                :  1, bin(8)           >
  <End MOVES Group>
<End CONTROL Group>
```

The structured field introducers produced for the LISTCONTROL STATUS subcommand with OUTPUT=FIELDS are:

```
<Begin CONTROL Group>
  <Begin STATUS Group>
    <STRM - DFSMSrmm status          :  1, bin(8)           >
    <JRNS - Journal status           :  1, bin(8)           >
    <STSL - Server listener status   :  1, bin(8)           >
    <STLO - Local tasks              :  3, bin(15)          >
    <STLA - Local active tasks       :  3, bin(15)          >
    <STLH - Local held tasks         :  3, bin(15)          >
    <STSO - Server tasks             :  3, bin(15)          >
    <STSA - Server active tasks      :  3, bin(15)          >
    <STSH - Server held tasks        :  3, bin(15)          >
    <STQR - Queued requests          :  4, bin(32)          >
    <STQN - Nowait requests          :  4, bin(32)          >
    <STQC - Catalog requests         :  4, bin(32)          >
    <STLR - Last RESERVE             :  4, packed decimal   >
    <STNH - New requests held        :  1, bin(8)           >
    <STRH - CDS reserved             :  1, bin(8)           >
    <STDS - Debug setting            :  1, bit(8)           >
    <STPL - Trace levels             :  1, bit(8)           >
  <End STATUS Group>
  <Begin TASKS Group>
    <STRF - Task req. function       :  5, character        >
    <STRT - Task req. system         :  8, character        >
```

```
    <STTR - Task req. type          :  3, character         >
    <STTQ - Task requestor          :  8, character         >
    <STST - Task start time         :  4, packed decimal    >
    <STTT - Task token              :  4, bin(32)           >
    <STTS - Task status             :  1, bin(8)            >
    <STIV - IP verb                 :  1, bin(8)            >
    <STIS - IP verb state           :  1, bin(8)            >
    <STIT - IP verb time            :  4, packed decimal    >
  <End TASKS Group>
<End CONTROL Group>
```

The structured field introducers produced for the LISTCONTROL DEFTABLE subcommand with OUTPUT=FIELDS are:

```
<Begin CONTROL Group>
   <Begin Default Group>
     <DEFD – DEFTABLE Data set name  : 44, character          >
     <DEFZ - DEFTABLE Continue       :  1, bin(8)            >
     <DEFJ – DEFTABLE Jobname        :  8, character         >
     <DEFN - DEFTABLE Program Name Ma:  8, character         >
     <DEFK – DEFTABLE Keydate        :  8, character         >
     <DEFE – DEFTABLE Retention Perio:  4, bin(31)           >
     <DEFM - DEFTABLE EDM            : 1, bin(8)             >
     <DEFP - DEFTABLE Scratch Pool   : 6, character          >
     <DEFO – DEFTABLE Retention Overr: 1, bin(8)             >
     <DEFV – DEFTABLE VRS Value      : 8, character          >
     <DEFL – DEFTABLE Last Reference : 4, bin(31)            >
     <DEFX – DEFTABLE VRSEL Excluded : 1, bin(8)             >
     <DEFR – DEFTABLE Retention Metho: 1, bin(8)             >
     <DEFY – DEFTABLE Retain By      : 1, bin(8)             >
     <DEFW – DEFTABLE WHILECATALOG   : 1, bin(8)             >
   <End Default Group>
<EndCONTROL Group>
```

When there is no information for a subgroup, such as MOVES, for the LISTCONTROL subcommand, the DFSMSrmm API returns all of the structured field introducers in the subgroup with no data. For example, when there are no outstanding volume actions, the DFSMSrmm API returns the MOVES subgroup (MFR, MST, MTO and MTY) with no data.

When DFSMSrmm cannot return all the output data for the LISTCONTROL subcommands in your output buffer, you must specify OPERATION=CONTINUE after processing your output buffer to obtain the rest of the LISTCONTROL output data.

Related reading: See “Using the CONTINUE operation in the EDGXCI macro” on page 23 for additional information.

LISTDATASET structured field introducers

The structured field introducers produced for the LISTDATASET subcommand with OUTPUT=FIELDS are:

```
<Begin DATASET Group>
  <DSN  - Data Set Name           : 44, character         >
  <CJBN - Job Name                :  8, character         >
  <VOL  - Volume Serial           :  6, character         >
  <OWN  - Owner                   :  8, character         >
  <DSEQ - Data Set Sequence       :  4, bin(31)           >
  <TZ   - Time Zone               :  4, bin(31)           >
  <DEV  - Device Number           :  4, character         >
  <FILE - Physical File Sequence  :  4, bin(31)           >
  <CDTJ - Create Date             :  4, packed decimal    >
  <CTM  - Create Time             :  4, packed decimal    >
  <SYS  - Creating system ID      :  8, character         >
  <BLKS - Block Size              :  4, bin(31)           >
  <BLKC - Block Count             :  4, bin(31)           >
  <LRCL - Logical Record Length   :  4, bin(31)           >
  <RCFM - Record Format           :  4, character         >
  <DC   - Data Class              :  8, character         >
  <DLWJ - Date Last Written       :  4, packed decimal    >
  <DLRJ - Date Last Read/Referenced:  4, packed decimal    >
  <STEP - Step Name               :  8, character         >
  <DD   - DD Name                 :  8, character         >
  <MC   - Management Class        :  8, character         >
  <SG   - Storage Group Name      :  8, character         >
  <SC   - Storage Class           :  8, character         >
  <VMV  - VRS Management Value    :  8, character         >
```

```
  <RTDJ - Retention Date           :  4, packed decimal    >
  <VTYP - Primary VRS Type         :  1, bin(8)            >
  <VJBN - Primary VRS Job Name     :  8, character         >
  <VNME - Primary VRS Name         : 44, character         >
  <VSCN - Primary VRS Subchain name:  8, character         >
  <VSCD - Primary VRS Subchain date:  4, packed decimal    >
  <VRSR - VRS Retained             :  1, bin(8)            >
  <NME  - Security Class Name      :  8, character         >
  <CLS  - Security Class Descriptio: 32, character         >
  <ABND - Abend while open         :  1, bin(8)            >
  <CTLG - Catalog status           :  1, bin(8)            >
  <2JBN - Secondary VRS jobname mas:  8, character         >
  <2NME - Secondary VRS mask       :  8, character         >
  <2SCN - Secondary VRS subchain na:  8, character         >
  <2SCD - Secondary VRS subchain da:  4, packed decimal    >
  <BLKT - Total block count        :  4, bin(31)           >
  <CPGM - Creating program name    :  8, character         >
  <LPGM - Last used program name   :  8, character         >
  <LJOB - Last used job            :  8, character         >
  <LSTP - Last used step name      :  8, character         >
  <LDD  - Last used DD name        :  8, character         >
  <LDEV - Last Drive               :  4, character         >
  <DPCT - Percent of volume        :  1, bin(8)            >
  <XDTJ - Expiration Date          :  4, packed decimal    >
  <XDSB - Expiry date set by       :  1, bin(8)            >
  <OXDJ - Original Expiration Date :  4, packed decimal    >
  <DSS6 - Data Set Size            : 14, compound          >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
  <DLTD - Deleted By Disposition Pr:  1, bin(8)            >
  <VEX  - VRSEL Exclude            :  1, bin(8)            >
  <BESK - CA Tape Encrytion key ind:  4, bin(31)           >
  <PSZ6 - Physical space used      : 14, compound          >
  <CRAT - Compression ratio        :  4, bin(31)           >
  <BLK6 - Total Block Count        :  8, bin(64)           >
  <LRED - LASTREF extra days       :  4, bin(31)           >
  <WCTL - WHILECATALOG for data set:  1, bin(8)            >
  <XTM  - Expiration Time          :  4, packed decimal    >
  <CTRT - Catalog Retained         :  1, bin(8)            >
  <FRXP - FORCEEXPIRE              :  1, bin(8)            >
<End DATASET Group>
```

LISTOWNER structured field introducers

The structured field introducers produced for the LISTOWNER subcommand with OUTPUT=FIELDS are:

```
<Begin OWNER Group>
  <OWN  - Owner                    :  8, character         >
  <SUR  - Owner's Surname          : 20, character         >
  <FOR  - Owner's Forename         : 20, character         >
  <DPT  - Owner's Department       : 40, character         >
  <ADL1 - Address Line 1           : 40, character         >
  <ADL2 - Address Line 2           : 40, character         >
  <ADL3 - Address Line 3           : 40, character         >
  <ITL  - Owner's Internal Tele Num:  8, character         >
  <ETL  - Owner's Ext Telephone Num: 20, character         >
  <EMU  - Owner's User ID          :  8, character         >
  <EMN  - Owner's Node             :  8, character         >
  <VLN  - Number of Volumes        :  4, bin(32)           >
  <EML  - Owner's Email Address    : 63, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
<End OWNER Group>
```

LISTPRODUCT structured field introducers

The structured field introducers produced for the LISTPRODUCT subcommand with OUTPUT=FIELDS are:

```
<Begin PRODUCT Group>
  <PNUM - Software Product Number  :  8, character         >
  <VER  - Software Product Version :  6, character         >
  <OWN  - Owner                    :  8, character         >
  <PNME - Product Software Name    : 30, character         >
  <PDSC - Product Description      : 32, character         >
  <VLN  - Number of Volumes        :  4, bin(32)           >
  <TZ   - Time Zone                :  4, bin(32)           >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
  <Begin PRODVOL Group>
    <VOL  - Volume Serial          :  6, character         >
    <RCK  - Rack or Bin Number     :  6, character         >
    <FCD  - Product Feature Code   :  4, character         >
  <End PRODVOL Group>
<End PRODUCT Group>
```

The PRODVOL group is repeated for each product volume.

LISTRACK structured field introducers

The structured field introducers produced for the LISTRACK subcommand with OUTPUT=FIELDS are:

```
<Begin RACK/BIN Group>
  <RCK  - Rack or Bin Number       :  6, character         >
  <VOL  - Volume Serial            :  6, character         >
  <RST  - Rack or Bin Status       :  1, bin(8)            >
  <LOC  - Location                 :  8, character         >
  <MEDN - Media Name               :  8, character         >
  <PID  - Pool Prefix              :  6, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
<End RACK/BIN Group>
```

LISTVOLUME structured field introducers

The structured field introducers produced for the LISTVOLUME subcommand with OUTPUT=FIELDS are:

```
<Begin VOLUME Group>
  <Begin VOL Group>
    <VOL  - Volume Serial            :  6, character         >
    <RCK  - Rack or Bin Number       :  6, character         >
    <OWN  - Owner                    :  8, character         >
    <TZ   - Time Zone                :  4, bin(31)           >
    <CJBN - Job Name                 :  8, character         >
    <CDTJ - Create Date              :  4, packed decimal    >
    <CTM  - Create Time              :  4, packed decimal    >
    <ADTJ - Assigned Date            :  4, packed decimal    >
    <ATM  - Assigned Time            :  4, packed decimal    >
    <XDTJ - Expiration Date          :  4, packed decimal    >
    <XDSB - Expiry date set by       :  1, bin(8)            >
    <OXDJ - Original Expiration Date :  4, packed decimal    >
    <RTDJ - Retention Date           :  4, packed decimal    >
    <DSN  - Data Set Name            : 44, character         >
    <VST  - Volume Status            :  1, bit(8)            >
    <OCE  - Volume Info. Recorded at :  1, bin(8)            >
    <AVL  - Volume Availability      :  1, bit(8)            >
    <LBL  - Volume Label             :  1, bit(8)            >
    <DEN  - Media Density            :  1, bin(8)            >
    <MDNF - Media Information Name   :  8, character         >
    <MEDT - Media Type               :  1, bin(8)            >
    <MDTX - External Media Type      :  8, character         >
```

```
    <MEDR - Media Recording Format   :  1, bin(8)           >
    <MDRX - External Recording Techn.:  8, character        >
    <MEDC - Media Compaction         :  1, bin(8)           >
    <MEDA - Media Special Attributes :  1, bin(8)           >
    <ACT  - Action on Release        :  1, bit(8)           >
    <PEND - Actions Pending          :  1, bit(8)           >
    <SG   - Storage Group Name       :  8, character        >
    <LOAN - Loan Location            :  8, character        >
    <ACN  - Account Number           : 40, character        >
    <DESC - Volume or VRS Description: 30, character        >
    <NME  - Security Class Name      :  8, character        >
    <CLS  - Security Class Descriptio: 32, character        >
    <VRSI - Scratch Immediate        :  1, bin(8)           >
    <VRXI - Expiration date ignore   :  1, bin(8)           >
    <VOLT - Volume Type              :  1, bin(8)           >
    <LVC  - Current label version    :  1, bin(8)           >
    <LVN  - Required label version   :  1, bin(8)           >
    <RBYS - Retain by set            :  1, bin(8)           >
    <STVC - Stacked volume count     :  4, bin(31)          >
    <SYS  - Creating system ID       :  8, character        >
    <DSYS - Creation System IDfirst f:  8, character        >
    <VOL1 - VOL1 label volser        :  6, character        >
    <WWID - Worldwide ID             : 24, character        >
    <VNDR - Vendor                   :  8, character        >
    <KEL1 - Encryption Key Label 1   : 64, character        >
    <KEL2 - Encryption Key Label 2   : 64, character        >
    <KEM1 - Encryption Encoding mech :  5, character        >
    <KEM2 - Encryption Encoding mech :  5, character        >
    <WORM - WORM flag                :  1, bin(8)           >
    <DKBC - Data sets kept by catalog:  4, bin(31)          >
    <XTM  - Expiration Time          :  4, packed decimal   >
    <CTRT - Catalog Retained         :  1, bin(8)           >
    <HLD  - Volume HOLD attribute    :  1, bin(8)           >
    <CRID - File 1 Create User ID    :  8, character        >
    <DSEQ - Data Set Sequence        :  4, bin(31)          >
    <OLON - Old Loan Location        :  8, character        >
    <RM   - Retention Method         :  1, bin(8)           >
    <RMSB - Retention Method Set By  :  1, bin(8)           >
    <EXRB - EXPDT RetainBy           : 1, bin(8)            >
    <EDM  - Managed externally       : 1, bin(8)            >
  <End VOL Group>
  <Begin ACCESS Group>
    <OAC  - Owner Access             :  1, bin(8)           >
    <VAC  - Volume Access            :  1, bin(8)           >
    <LCID - Last Change User ID      :  8, character        >
    <VM   - VM Use                   :  1, bin(8)           >
    <MVS  - MVS Use                  :  1, bin(8)           >
    <IRMM - IRMM use                 :  1, bin(8)           >
    <LCDJ - Last Change Date         :  4, packed decimal   >
    <LCTM - Last Change Time         :  4, packed decimal   >
    <LCSI - Last Change System ID    :  8, character        >
    <LCUD - Last "User" Change Date  :  4, packed decimal   >
    <LCUT - Last "User" Change Time  :  4, packed decimal   >
    <UID01- User ID 1                :  8, character        >
  <End ACCESS Group>
  <Begin STAT Group>
    <DSC  - Data Set Count           :  4, bin(31)          >
    <DSR  - Data Set Recording       :  1, bin(8)           >
    <USEM - Volume Usage (KB)        :  4, bin(31)          >
    <USEC - Volume Use Count         :  4, bin(31)          >
    <DLRJ - Date Last Read/Referenced:  4, packed decimal   >
    <DLWJ - Date Last Written        :  4, packed decimal   >
    <LDEV - Last Drive               :  4, character        >
    <SEQ  - Volume Sequence          :  4, bin(31)          >
    <MEDN - Media Name               :  8, character        >
    <PVL  - Previous Volume          :  6, character        >
    <NVL  - Next Volume              :  6, character        >
    <PNUM - Software Product Number  :  8, character        >
    <VER  - Software Product Version :  6, character        >
    <FCD  - Product Feature Code     :  4, character        >
    <TRD  - Temporary Read Errors    :  4, bin(31)          >
    <TWT  - Temporary Write Errors   :  4, bin(31)          >
    <PRD  - Permanent Read Errors    :  4, bin(31)          >
    <PWT  - Permanent Write Errors   :  4, bin(31)          >
    <VCAP - Volume capacity          :  4, bin(31)          >
    <VPCT - Volume percent full      :  1, bin(8)           >
    <VWMC - Volume Write Mount Count :  4, bin(31)          >
    <USE6 - Volume Usage             : 14, compound         >
    <PSZ6 - Physical space used      : 14, compound         >
    <CRAT - Compression ratio in hund:  6, bin(31)          >
  <End STAT Group>
  <Begin STORE Group>
```

```
    <LOC  - Location                 :  8, character         >
    <LOCT - Location Type            :  1, bin(8)            >

    <DEST - Destination Name         :  8, character         >
    <DSTT - Destination Type         :  1, bin(8)            >

    <INTR - Volume Intransit Status  :  1, bin(8)            >
    <HLOC - Home Location            :  8, character         >
    <HLOT - Home Location Type       :  1, bin(8)            >

    <OLOC - Old Location             :  8, character         >
    <OLOT - Old Location Type        :  1, bin(8)            >

    <NLOC - Required Location        :  8, character         >
    <NLOT - Required Location Type   :  1, bin(8)            >

    <SDTJ - Movement Tracking Date   :  4, packed decimal    >
    <MOVM - Move Mode                :  1, bin(8)            >
    <BIN  - Bin Number               :  6, character         >
    <BMN  - Bin Number Media Name    :  8, character         >
    <OBN  - Old Bin Number           :  6, character         >
    <OBMN - Old Bin Number Media Name:  8, character         >
    <CTNR - Container                : 16, character         >
    <DBIN - Destination Bin number   :  6, character         >
    <DBMN - Destination Bin media nam:  8, character         >
    <RLPR - Required Location Priorit:  4, bin(31)           >
  <End STORE Group>
<End VOLUME Group>
```

LISTVRS structured field introducers

The structured field introducers produced for the LISTVRS subcommand with OUTPUT=FIELDS are:

```
<Begin VRS Group>
  <VRS  - Vital Record Specificatio: 44, character         >
  <TYP  - VRS Type                 :  1, bit(8)            >
  <VJBN - Primary VRS Job Name     :  8, character         >
  <VRC  - Vital Record Count       :  4, bin(32)           >
  <RET  - Retention Type           :  3, bin(8)            >
  <VDD  - VRS Delay Days           :  2, bin(15)           >
  <LOC  - Location                 :  8, character         >
  <SC1  - Store Number             :  4, bin(32)           >
  <PRTY - Priority                 :  4, bin(32)           >
  <NVRS - Next VRS Name            :  8, character         >
  <OWN  - Owner                    :  8, character         >
  <DESC - Volume or VRS Description: 30, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <DDTJ - Delete/Store Date        :  4, packed decimal    >
  <VANX - Next VRS Type            :  1, bin(8)            >
  <VRSI - Scratch Immediate        :  1, bin(8)            >
  <VRXI - Expiration date ignore   :  1, bin(8)            >
  <DLRJ - Date Last Read/Referenced:  4, packed decimal    >
  <TLR  - Time Last Read/Referenced:  4, packed decimal    >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
<End VRS Group>
```

SEARCH-Type of subcommands

The DFSMSrmm SEARCH-type of subcommands are: SEARCHBIN, SEARCHDATASET, SEARCHOWNER, SEARCHPRODUCT, SEARCHRACK, SEARCHVOLUME, and SEARCHVRS. You use these subcommands to obtain information from the DFSMSrmm control data set about resources defined to DFSMSrmm.

When you specify OUTPUT=FIELDS, the DFSMSrmm API returns data for all SEARCH type of subcommands as structured fields. DFSMSrmm returns the output data for one or more resources in your output buffer each time you call the API. Use the MULTI=YES keyword to specify that your application can handle multiple resources returned in your output buffer. You must specify OPERATION=CONTINUE after processing your output buffer to obtain the output data for the next resource or set of resources. Continue to call the DFSMSrmm API until the output data for all matching resources has been returned.

Related Reading: See “Using the CONTINUE operation in the EDGXCI macro” on page 23 for additional information.

The DFSMSrmm API returns expanded output data for the RMM TSO SEARCHDATASET, SEARCHPRODUCT, SEARCHVOLUME, and SEARCHVRS subcommands when you also specify the EXPAND=YES parameter.

SEARCHBIN structured field introducers

The output that DFSMSrmm returns when you specify the SEARCHBIN subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin RACK/BIN Group>
  <RCK  - Rack or Bin Number       :  6, character         >
  <VOL  - Volume Serial            :  6, character         >
  <RST  - Rack or Bin Status       :  1, bin(8)            >
  <LOC  - Location                 :  8, character         >
  <MEDN - Media Name               :  8, character         >
  <MIV  - Moving-In Volume         :  6, character         >
  <MOV  - Moving-Out Volume        :  6, character         >
  <OVOL - Old Volume               :  6, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
<End RACK/BIN Group>
```

SEARCHDATASET structured field introducers

The output DFSMSrmm returns when you specify the SEARCHDATASET subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin DATASET Group>
  <DSN  - Data Set Name            : 44, character         >
  <VOL  - Volume Serial            :  6, character         >
  <OWN  - Owner                    :  8, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <CDTJ - Create Date              :  4, packed decimal    >
  <CTM  - Create Time              :  4, packed decimal    >
  <FILE - Physical File Sequence   :  4, bin(32)           >
  <RTDJ - Retention Date           :  4, packed decimal    >
  <XDTJ - Expiration Date          :  4, packed decimal    >
  <CTRT - Catalog Retained         : 13, character         >
  <DSGC - Total GDG generation count: 4, bin(32)           >
  <DGSN - Relative GDG generation number: 4,bin(32)        >
<End DATASET Group>
```

The expanded output that DFSMSrmm returns when you specify the SEARCHDATASET subcommand with the OUTPUT=FIELDS and EXPAND=YES parameters is the same as shown in “LISTDATASET structured field introducers” on page 51 for LISTDATASET.

SEARCHOWNER structured field introducers

The output DFSMSrmm returns when you specify the SEARCHOWNER subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin OWNER group>
  <OWN  - owner                    : 8  , character:         >
  <SUR  - owner's surname          : 20 , character:         >
  <FOR  - owner's forename         : 20 , character:         >
  <DPT  - owner's department       : 40 , character:         >
  <ADL  - address line             : 40 , character:         >
  <ADL  - address line             : 40 , character:         >
  <ADL  - address line             : 40 , character:         >
  <ITL  - owner's internal tel num : 8  , character:         >
  <ETL  - owner's external tele num: 20 , character:         >
  <EMU  - owner's user ID          : 8  , character:         >
  <EMN  - owner's node             : 8  , character:         >
  <VLN  - number of volumes        : 4  , bin(32):           >
  <EML  - owner's email address    : 63 , character:         >
  <TZ   - time zone                : 4  , bin(32):           >
<End OWNER group>
```

SEARCHPRODUCT structured field introducers

The output DFSMSrmm returns when you specify the SEARCHPRODUCT subcommand and the EDGXCI macro OUTPUT=FIELDS parameter is:

```
<Begin PRODUCT Group>
  <PNUM - Software Product Number  :  8, character         >
  <VER  - Software Product Version :  6, character         >
  <OWN  - Owner                    :  8, character         >
  <PNME - Product Software Name    : 30, character         >
  <PDSC - Product Description      : 32, character         >
  <VLN  - Number of Volumes        :  4, bin(32)           >
  <TZ   - Time Zone                :  4, bin(32)           >
  <LCDJ - Last Change Date         :  4, packed decimal    >
  <LCTM - Last Change Time         :  4, packed decimal    >
  <LCID - Last Change User ID      :  8, character         >
  <LCSI - Last Change System ID    :  8, character         >
  <LCUD - Last "User" Change Date  :  4, packed decimal    >
  <LCUT - Last "User" Change Time  :  4, packed decimal    >
  <Begin PRODVOL Group>
    <VOL  - Volume Serial          :  6, character         >
    <RCK  - Rack or Bin Number     :  6, character         >
    <FCD  - Product Feature Code   :  4, character         >
  <End PRODVOL Group>
<End PRODUCT Group>
```

EXPAND=NO and EXPAND=YES return the same data elements so the EXPAND parameter can be omitted. Unlike LISTPRODUCT the SEARCHPRODUCT command returns only the PRODVOL group for the first product volume, if at least one volume exists.

SEARCHRACK structured field introducers

The output DFSMSrmm returns when you specify the SEARCHRACK subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin RACK/BIN Group>
  <RCK  - Rack or Bin Number       :  6, character         >
  <VOL  - Volume Serial            :  6, character         >
  <RST  - Rack or Bin Status       :  1, bin(8)            >
  <LOC  - Location                 :  8, character         >
  <MEDN - Media Name               :  8, character         >
  <PID  - Pool Prefix              :  6, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
<End RACK/BIN Group>
```

SEARCHVOLUME structured field introducers

The output DFSMSrmm returns when you specify the SEARCHVOLUME subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin VOLUME Group>
  <VOL  - Volume Serial            :  6, character         >
  <OWN  - Owner                    :  8, character         >
  <RCK  - Rack or Bin Number       :  6, character         >
  <TZ   - Time Zone                :  4, bin(32)           >
  <ADTJ - Assigned Date            :  4, packed decimal    >
  <XDTJ - Expiration Date          :  4, packed decimal    >
  <RTDJ - Retention Date           :  4, packed decimal    >
  <LOC  - Location                 :  8, character         >
  <INTR - Volume Intransit Status  :  1, bin(8)            >
  <HLOC - Home Location            :  8, character         >
  <DSC  - Data Set Count           :  4, bin(32)           >
  <VST  - Volume Status            :  1, bit(8)            >
  <AVL  - Volume Availability      :  1, bit(8)            >
  <LBL  - Volume Label             :  1, bit(8)            >
  <MEDT - Media Type               :  1, bin(8)            >
  <MEDR - Media Recording Format   :  1, bin(8)            >
  <MEDC - Media Compaction         :  1, bin(8)            >
  <MEDA - Media Special Attributes :  1, bin(8)            >
  <PEND - Actions Pending          :  1, bit(8)            >
  <LOAN - Loan Location            :  8, character         >
  <DEST - Destination Name         :  8, character         >
  <DSR  - Data Set Recording       :  1, bin(8)            >
```

```
  <SEQ  - Volume Sequence          :  4, bin(32)          >
  <MEDN - Media Name               :  8, character        >
  <LVC  - Current label version    :  1, bin(8)           >
  <LVN  - Required label version   :  1, bin(8)           >
<End VOLUME Group>
```

The expanded output that DFSMSrmm returns when you specify the SEARCHVOLUME subcommand with the OUTPUT=FIELDS and EXPAND=YES parameters is the same as shown in "LISTVOLUME structured field introducers" on page 53 for LISTVOLUME.

SEARCHVRS structured field introducers

The output DFSMSrmm returns when you specify the SEARCHVRS subcommand and the EDGXCI macro OUTPUT=FIELDS and EXPAND=NO parameters is:

```
<Begin VRS Group>
  <VRS  - Vital Record Specificatio: 44, character        >
  <TYP  - VRS Type                 :  1, bit(8)           >
  <VJBN - Primary VRS Job Name     :  8, character        >
  <RET  - Retention Type           :  3, bin(8)           >
  <LOC  - Location                 :  8, character        >
  <PRTY - Priority                 :  4, bin(32)          >
  <NVRS - Next VRS Name            :  8, character        >
  <OWN  - Owner                    :  8, character        >
  <TZ   - Time Zone                :  4, bin(32)          >
  <DDTJ - Delete/Store Date        :  4, packed decimal   >
  <VANX - Next VRS Type            :  1, bin(8)           >
  <VRSI - Scratch Immediate        :  1, bin(8)           >
  <VRXI - Expiration date ignore   :  1, bin(8)           >
  <VRC  - Vital Record Count       :  4, bin(32)          >
  <SC1  - Store Number             :  4, bin(32)          >
  <DLRJ - Date Last Read/Referenced:  4, packed decimal   >
  <TLR  - Time Last Read/Referenced:  4, packed decimal   >
<End VRS Group>
```

The expanded output that DFSMSrmm returns when you specify the SEARCHVRS subcommand with the OUTPUT=FIELDS and EXPAND=YES parameters is the same as shown in "LISTVRS structured field introducers" on page 55 for LISTVRS.

Controlling output from list and search type requests

The DFSMSrmm API returns information for a SEARCH type of subcommand or for a LISTCONTROL subcommand based on these factors:

- Whether you want line format or field format data.
- Whether you want one or multiple resources in your output buffer
- The size of your output buffer.
- The amount of output data.
- The LIMIT operand value used for a SEARCH type of subcommand.

Limiting the search for a request

Use the LIMIT keyword on SEARCH type of subcommands to limit the number of entries DFSMSrmm returns. To conserve use of system resources, such as dynamic storage, DFSMSrmm suspends a search operation after the number of entries matches the limit value you specify or the default limit value.

When you issue an RMM TSO Search type of subcommand, you can use the LIMIT operand to limit the number of entries returned. DFSMSrmm ends the search because the limit you set is reached or all available entries have been returned.

For an application program, the DFSMSrmm API causes DFSMSrmm to resume the search. LIMIT does not limit the total number of entries that the DFSMSrmm API returns to your application program and you cannot use LIMIT to end the subcommand before you have received all of the entries for a subcommand.

Instead, you can specify OPERATION=CONTINUE regardless of whether limit has been reached, or begin a new command, or use EDGXCI OPERATION=RELEASE.

Output buffer examples

The examples in this section illustrate the:

- SEARCH type subcommands (and LISTCONTROL) might require your application program to use one or more OPERATION=CONTINUE calls to the DFSMSrmm API to receive all of the search results.
- Your application program should expect to receive more than one set of return and reason codes. In the example, DFSMSrmm issued a different set of codes for each output buffer:
  - Return code 0, reason code 4.
  - Return code 4, reason code 2.
  - Return code 4, reason code 4.

  Depending on the subcommand that you specify, the search criteria that you specify (fully or partially qualified names), and whether you specify a LIMIT value or LIMIT(*), DFSMSrmm can also issue these return codes and reason codes.
  - Return code 0, reason code 0.
  - Return code 4, reason code 8.

  For more information about the return codes and reason codes that the API returns, see Table 9 on page 24.
- Header lines for search lists are placed at the beginning of the first output buffer of each set of buffers: The first output buffer after OPERATION=BEGIN, and the first output buffer after OPERATION=CONTINUE in response to the return code 4 and reason code 2.
- Messages issued by DFSMSrmm and that are placed in your output buffers are introduced by <MSGL> structured field introducers rather than <LINE> structured field introducers.
- The number of output data lines that are placed in your buffer is dependent upon the interaction of:
  - The total number of searched records (entries).
  - The size of your output buffer.
  - The LIMIT value used for the search.

Figure 32 on page 60, Figure 33 on page 60, and Figure 34 on page 61 display the contents of the output buffers when:

- Your application program issues an OPERATION=BEGIN, OUTPUT=LINES for a SEARCHRACK RACK(*) LIMIT(90) subcommand.
- Your application program is using a minimum size (4096 bytes) output buffer.
- There are 130 records in the RMM inventory.

First output buffer

The DFSMSrmm API issues return code 0 and reason code 4 and returns control to your application program. Your output buffer contains 78 structured fields.

In Figure 32 on page 60:

- The group begins with the <Begin RACK or BIN group>.
- The structured fields between the Begin and End RACK group structured field introducers are all introduced by a <LINE> SFI.
- The first two lines after the Begin RACK group are the header lines for the list of RACK entries.
- The group ends with the <End RACK or BIN group>.

The DFSMSrmm API returns code 0 and reason code 4 when there is more output data. Specify the EDGXCI macro OPERATION=CONTINUE parameter to continue the subcommand request..

```
<Begin RACK or BIN group>
  <LINE>Rack    Medianame  Volume  Status    Location
  <LINE>------  ---------  ------  --------  --------
  <LINE>020610  CART3480   020610  IN USE    SHELF
  <LINE>020742  CART3480   020742  IN USE    SHELF
  <LINE>021042  CART3480   021042  IN USE    SHELF
    ...
    ...
  <LINE>030311  CART3480   030311  IN USE    SHELF
  <LINE>030318  CART3480   030318  IN USE    SHELF
<End RACK or BIN group>
```

Figure 32. CONTINUE example, first output buffer

Second output buffer

After processing the OPERATION=CONTINUE parameter, the DFSMSrmm API continues processing. The DFSMSrmm API issues return code 4 and reason code 2, returns control to your application program. Your output buffer contains 20 structured fields.

```
<Begin RACK or BIN group>
  <LINE>031086  CART3480   031086  IN USE    SHELF
  <LINE>031568  CART3480   031568  IN USE    SHELF
  <LINE>031599  CART3480   031599  IN USE    TRON
    ...
    ...
  <LINE>032848  CART3480   032848  IN USE    SHELF
  <LINE>032898  CART3480   032898  IN USE    SHELF
  <MSGL>EDG3203I SEARCH COMPLETE - MORE ENTRIES MAY EXIST
  <MSGL>EDG3012I 90         ENTRIES LISTED
<End RACK or BIN group>
```

Figure 33. CONTINUE example, second output buffer

In Figure 33 on page 60:

- There are no header lines in the second output buffer.
- There are only 16 output data lines (the LINE structured field introducers).
- The last output data line is followed by two message lines introduced by the <MSGL> SFI.

The DFSMSrmm API returns control to your application program even though there is room in the output buffer for more data. This is because the LIMIT value of 90 was reached as indicated by the second message line.

The return code 4 and reason code 2 indicate that more entries might exist. When you use OPERATION=CONTINUE, one of these statements is likely to occur:

- When there are more entries, your application program receives control back with more output data in your output buffer.
- When there are no other entries, your application program receives control back with a buffer that is empty or that contains only messages.

Third (Last) output buffer

After the second OPERATION=CONTINUE, control is returned to your application program with return code 4 and reason code 4, and your output buffer contains 45 structured fields.

```
<Begin RACK or BIN group>
  <LINE>Rack    Medianame  Volume  Status    Location
  <LINE>------  ---------  ------  --------  --------
  <LINE>032935  CART3480   032935  IN USE    SHELF
  <LINE>032941  CART3480   032941  IN USE    SHELF
  <LINE>032946  CART3480   032946  IN USE    SHELF
    ...
    ...
  <LINE>070692  CART3480   070692  IN USE    SHELF
  <LINE>070693  CART3480   070693  IN USE    SHELF
  <MSGL>EDG3012I 40         ENTRIES LISTED
<End RACK or BIN group>
```

Figure 34. CONTINUE example, third (Last) output buffer

In Figure 34 on page 61:

- The first two lines after the Begin RACK group are the header lines that you saw in the first output buffer. This is the output for a second search that the DFSMSrmm API started when you specified OPERATION=CONTINUE in response to the return code 4 and reason code 2.
- The last output data line in your output buffer is followed by a single message line.
- The return code 4 and reason code 4 indicate that the subcommand was ended before the LIMIT value was reached.
- The total number of entries given to your application program in the three output buffers is 130: 74 in the first, 16 in the second, and 40 in the last output buffer.

Appendix A. Structured field introducers (SFIs)

This section defines the structured field introducers (SFIs) used by the DFSMSrmm API to identify fields in API output.

Structured field introducer (SFI) format

All structured field introducers have this format:

Bytes
Description

0-1
2-byte length: SFI length plus data length

2-4
3-byte identifier: SFI ID (hexadecimal)

5
1-byte type modifier: Type of SFI

- 0 = 8-byte, fixed-length SFI

6
1-byte (Reserved)

7
1-byte data type: Type of data, if any, that follows the SFI

- 0=Undefined (no data)
- 1=Character (fixed-length)
- 2=Bit(8) (1-byte flag, multiple bits can be on)
- 3=Binary(8) (1-byte (hex) value)
- 4=Binary(15) (2-byte (hex) value)
- 5=Binary(32) (4-byte (hex) unsigned value)
- 6=Binary(64) (8-byte (hex) value)
- 7=Character (variable-length)
- 8=Compound SFI (multiple related values, see “Compound SFI” on page 64.)
- 9=(4 bytes) Packed decimal Julian date: yyyydddC
- A=(4 bytes) Packed decimal time format: hhmmsstC

Structured field lengths

All structured fields have a minimum length of 8 bytes (for the structured field introducer). The length can be fixed-length or variable-length.

- **Fixed-length**:

  The structured field has one of two length values: 8 when there is no data or the defined maximum length. For example, if the length is defined as X'000C' (decimal 12) for a particular structured field, the length in the structured field introducer has a value of either X'0008' (no data) or X'000C' (data length = 4).

- **Variable-length**:

  The structured field can have a length that varies from 8 (no data) up to maximum stated size. For example, because a data set name varies from 1 to 44 characters in length, the length value in a

structured field introducer for a data set name can be X'0008' (no data), or it can vary from X'0009' to X'0034' (9 to 52 decimal).

Compound SFI

A compound SFI includes multiple values each with own data type and length.

Compound type:

1
Factored. A Binary(8) value combined with a second field containing a count. The second field is identified by a data type.

Factor values:

0
Bytes (unfactored)

1
KB

2
MB

3
GB

4
TB

and so on.

Compound structured field introducers follow this structure;

Byte Count
Description

8
Standard SFI including 1 byte data type identifier (X'08')

1
Compound type identifier; 1 = Factored; 2 self describing fields where the first is the factor used, and the second is the resultant value

1
Length of the first field, including this byte

1
Data type identifier

n
First data field as identified by the preceding data type field; for example Binary(8)

1
Length of the next field, including this byte

1
Data type identifier

n
Next data field as identified by the preceding data type field; for example Binary(64)

Structured field introducers for Begin and End Resource groups

Begin and End Resource group structured field introducers identify when the output for a particular resource begins and ends. Begin and End Resource groups can be used to identify subgroups within a group. The Begin and End Resource groups are never followed by data. Table 13 on page 65 shows structured field introducers that identify Begin and End resource groups.

Table 13. Begin and End Resource group structured field introducers

Begin - End IDs	Resource Group
X'021000' - X'021080'	ACCESS - within VOLUME
X'022000' - X'022080'	ACTIONS - within CONTROL
X'024000' - X'024080'	CNTL - within CONTROL
X'025000' - X'025080'	CONTROL
X'026000' - X'026080'	DATASET
X'026500' - X'026580'	DEFAULT - within CONTROL
X'027000' - X'027080'	LOCDEF - within CONTROL
X'027500' - X'027580'	MEDINF - within CONTROL
X'028000' - X'028080'	MESSAGE
X'029000' - X'029080'	MNTMSG - within CONTROL
X'02A000' - X'02A080'	MOVES - within CONTROL
X'03A000' - X'03A080'	OPENRULE within CONTROL
X'02B000' - X'02B080'	OPTION - within CONTROL
X'02C000' - X'02C080'	OWNER
X'02D000' - X'02D080'	PRODUCT
X'039000' - X'039080'	PRODVOL - within PRODUCT
X'03B000' - X'03B080'	PRTITION within CONTROL
X'02E000' - X'02E080'	RACK or BIN
X'02F000' - X'02F080'	REJECT - within CONTROL
X'030000' - X'030080'	SECCLS - within CONTROL
X'031000' - X'031080'	SECLVL - within CONTROL
X'032000' - X'032080'	STAT - within VOLUME
X'033000' - X'033080'	STORE - within VOLUME
X'034000' - X'034080'	SYSRETC
X'035000' - X'035080'	VLPOOL - within CONTROL
X'036000' - X'036080'	VOL - within VOLUME
X'037000' - X'037080'	VOLUME
X'038000' - X'038080'	VRS
X'03C000' - X'03C080'	STATUS - within CONTROL
X'03D000' - X'03D080'	TASKS – within CONTROL

Structured field introducers for return and reason codes

The structured field introducers shown in Table 14 on page 66 provide return codes and reason codes in your output buffer.

The DFSMSrmm API issues the return and reason code structured field introducers only when the subcommand fails. Each return and reason code pair is grouped within the SYSRETC group. The FRC and FRS structured field introducers are used for return and reason codes that are returned from OAM.

The RSNC and RTNC structured field introducers are used for return and reason codes that are from another system service.

When the DFSMSrmm API builds a SYSRETC group for an error reported by a system service, look for additional information that is available from system messages in places like the operator terminal, SYSTSPRT, job log, and SYSLOG data set.

Subcommands are described using standard DFSMSrmm abbreviations. For example, AV is for ADDVOLUME as shown in Table 3 on page 1. The structured field introducer values are enclosed in single quotes (') to signify that they are 8-byte hexadecimal values. Two spaces are included in the IDs for readability.

Table 14. Reason and return code structured field introducers

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'400000'	FRC	12	Binary(32)	Function return code	AV CV DV GV
X'401000'	FRS	12	Binary(32)	Function reason code	AV CV DV GV
X'402000'	RSNC	12	Binary(32)	Reason code	Any subcommand
X'403000'	RTNC	12	Binary(32)	Return code	Any subcommand
X'404000'	SVCN	16	Character (variable length)	Service name	Any subcommand

Structured field introducers for messages and message variables

The structured field introducers described in Table 15 on page 66 introduce messages and message variables that the DFSMSrmm API places in your output buffer:

- MSGL is used when OUTPUT=LINES.
- MSGN and ENTN are used when OUTPUT=FIELDS.
- The SFI definitions are enclosed in single quotes (') to signify that they are 8-byte values and the two spaces are inserted for readability.

The MSGN and ENTN structured field introducers are always grouped within the MESSAGE group. The MSGL structured field introducers are grouped within the MESSAGE group when the DFSMSrmm API is unable to determine which subcommand type the message is for. One or more structured field introducers other than ENTN might follow MSGN as described in “Messages and message variables structured field introducers” on page 43.

Table 15. Message structured field introducers

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'051000'	MSGL	259	Character (variable length)	Message line	Any subcommand
X'052000'	MSGN	16	Character (fixed length)	Message number ID	As previously defined
X'053000'	ENTN	12	Binary(32)	Number of entries Min 0, Max 10-digit	As previously defined
X'054000'	KEYF	65	Character (variable length)	Key from	SD SV
X'054200'	KEYT	65	Character (variable length)	Key to	SD SV
X'055000'	TYPF	16	Character (variable length)	VOLUME or DATASET	SD SV

Table 15. Message structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'055200'	TYPT	16	Character (variable length)	VOLUME or DATASET	SD SV
X'057000'	CONT	92	Character (variable length)	SEARCH Continue information	All search subcommands

Structured field introducers for subcommand output data

The structured field introducers described in Table 16 on page 67 introduce subcommand output data in your output buffer. These structured field introducers are always grouped within a pair of Begin and End Resource group structured field introducers.

This notation is used:

- Subcommands are described using standard DFSMSrmm abbreviations. For example, LV is for LISTVOLUME and SS is for SEARCHVRS as described in Table 3 on page 1.
- The (e) following a search type of subcommand abbreviation means the expanded output is available if you specify EXPAND=YES. The absence of (e) means the SFI is used for both EXPAND=NO and EXPAND=YES.
- The range of two-byte and four-byte numbers is denoted by the minimum expected value and the maximum number of digits the number is expected to have. For example: "Min 1, Max 4-digit" means the minimum expected value of the number is one and the maximum expected number of digits in the number is four.
- The SFI definitions are enclosed in single quotes (') to signify that they are 8-byte values and the two spaces are inserted for readability. Bit data (flags) values are also enclosed in single quotes.

Table 16. Command structured field introducers

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8C6100					
X'800500'	ABND	9	Binary(8)	Closed by Abend 0=NO 1=YES	LD, SD(e)
X'800800'	ACCT	9	Binary(8)	Accounting source 0=JOB 1=STEP	LC
X'801000'	ACN	48	Character (variable length)	Account number	LV, SV(e)
X'801800'	ACS	9	Binary(8)	SMSACS 0=NO 1=YES	LC
X'802000'	ACT	9	Bit(8)	Actions on release '80'=SCRATCH '40'=REPLACE '20'=INIT '10'=ERASE '08'=RETURN '04'=NOTIFY For LC VLPOOL X'00', X'04'	LC, LV, SV(e)
X'803001'	ADL	48	Character (variable length)	Address line. The SFI is incremented by one for each ADL line that is found. (X'803001' - X'803003')	LO, SO
X'804000'	ADTJ	12	Packed decimal Julian date format	Assigned date	LV, SV
X'805000'	AST	9	Bit(8)	Action status '80'=PENDING '40'=CONFIRMED '20'=COMPLETE '10'=UNKNOWN	LC
X'806000'	ATM	12	Packed decimal time format	Assigned time	LV, SV(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'807000'	AUD	10	Binary(15)	SMF audit record type: 128-255, 42, or 0	LC
X'808000'	AVL	9	Bit(8)	Volume availability '40'=PENDING_RELEASE '20'=VITAL_RECORD '08'=ON_LOAN '04'=OPEN	LV, SV
X'809000'	BDTJ	12	Packed decimal Julian date format	Last control data set backup date	LC
X'809310'	BESK	12	Binary(32)	CA Tape Encryption key index, 4 byte hex value	LD, SD(e)
X'80A000'	BIN	14	Character (fixed length)	6-character alphanumeric bin number	LV, SV(e)
X'80B000'	BKPP	16	Character (Variable length)	Backup procedure name	LC
X'80C000'	BLKC	12	Binary(32)	Block count	LD, SD(e)
X'80D000'	BLKS	12	Binary(32)	Block size	LD, SD(e)
X'80D030'	BLKT	12	Binary(32)	Total block count	LD, SD(e)
X'80D0B0'	BLK6	16	Binary(64)	Total block count	LD
X'80E000'	BLP	9	Binary(8)	BLP option: 0=RMM 1=NORMM	LC
X'80F000'	BMN	16	Character (variable length)	Bin number media name	LV, SV(e)
X'810000'	BTM	12	Packed decimal time format	Last control data set backup time	LC
X'811000'	CACT	9	Bit(8)	Control active functions '80'=BACKUP '40'=RESTORE '20'=VERIFY '10'=EXPROC '08'=EXTRACT '04'=DSTORE '02'=VRSEL	LC
X'811800'	CATS	9	Binary(8)	CATSYSID value 0=SET 1=NOTSET 2=*	LC
X'812000'	CDS	16	Character (variable length)	Control data set identifier	LC
X'812400'	CDSF	10	Binary(15)	CDSFULL parmlib value: 0-99	LC
X'812900'	CDSQ	9	Binary(8)	Control data set ENQ 0=Disabled 1=Enabled	LC
X'812A00'	CDSU	10	Binary(15)	Control data set percentage used	LC
X'813000'	CDTJ	12	Packed decimal Julian date format	Create date	LD, LV, SD, SV(e)
X'814000'	CJBN	16	Character (variable length)	Job name	LD, LV, SD(e), SV(e)
X'815000'	CLIB	16	Character (variable length)	Current library name	AV, CV, DV
X'816000'	CLS	40	Character (variable length)	Security class description	LC, LD, LV, SD(e), SV(e)
X'816900'	CMDD	9	Binary(8)	Command Authorization - based on DSN: 0=No 1=Yes	LC
X'8169A0'	CMDO	9	Binary(8)	Command Authorization - based on owner: 0=No 1=Yes	LC
X'817000'	CNT	12	Binary(32)	Bin, rack, or volume count: Min 0, Max 5-digit	AB, AR, AV, DB, DR
X'817820'	CPGM	16	Character (fixed length)	Creating program name	LD, SD(e)
X'817890'	CRAT	12	Binary(32)	Compression ratio in hundreths	LD, LV
X'817900'	CRID	16	Character (variable length)	File 1 create user ID	LV VOL, SV(e)
X'818000'	CRP	12	Binary(32)	CATRETPD retention period: Min 0 Max 4-digit	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'818800'	CSDT	12	Packed decimal Julian date	Catalog synchronize date	LC
X'819000'	CSG	16	Character (variable length)	Current storage group name	AV, CV
X'819200'	CSHN	71	Character (variable length)	Client/server host name 1-to-63 alphanumeric characters including hyphen, period, and blank	LC
X'819250'	CSIP	53	Character (variable length)	Client IP address 1-to-45 numeric characters including colon, period, and blank	LC
X'819400'	CSTM	12	Packed decimal time date	Catalog synchronize time	LC
X'819600'	CSVE	9	Binary(8)	Stacked volume enable status: 0=None 1=Enabled 2=Disabled 3=Mixed	LC
X'819785'	CTLD	12	Binary(32)	Catalog Days Min 0, Max 93000	LC
X'819800'	CTLG	9	Binary(8)	Catalog status: 0=UNKNOWN 1=NO 2=YES	LD, SD(e)
X'81A000'	CTM	12	Packed decimal time format	Create time	LD, LV, SD, SV(e)
X'81A300'	CTNR	24	Character (variable length)	In container	LV STORE
X'81A400'	CTRT	9	Binary(8)	CatalogRetained status for data set or volume: data set 0 = OFF 1 = KeptByCatalog 2 = UntilExpired 3 = ForceExpire	LD, LV, SD(e), SV(e)
X'81A600'	DBIN	14	Character (fixed length)	Numeric: 0–999999 or 6 alphanumeric character destination bin number	LV
X'81A700'	DBMN	16	Character (variable length)	Destination bin media name	LV
X'81B000'	DBN	12	Binary(32)	Bin numbers in DISTANT location: Min 0, Max 6-digit	LC
X'81C000'	DC	16	Character (variable length)	Data class name	LD, SD(e)
X'81D000'	DD	16	Character (variable length)	DD name	LD, SD(e)
X'81E000'	DDTJ	12	Packed decimal Julian date format	Delete date or last store update date	LC, LS, SS
X'81E050'	DEFC	10	Binary(15)	Defaults table entries count	LC OPT
X'81E100'	DEFD	52	Character (variable length)	Defaults table data set name mask ** = all DSNs match	LC DEFTABLE
X'81E200'	DEFE	12	Binary(32)	Defaults table retention period default Min 0, Max 93000 FFFFFFFE=permanent FFFFFFFD=blank	LC DEFTABLE
X'81E300'	DEFJ	16	Character (variable length)	Defaults table jobname mask * = all jobnames match	LC DEFTABLE
X'81E400'	DEFK	17	Character (variable length)	Defaults table keydate mask 5 characters of format yyddd or NOKEYDATE * = all dates match	LC DEFTABLE

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'81E500'	DEFL	12	Binary(32)	Defaults table last reference extra days default Min 0, Max 93000 FFFFFFFD=blank	LC DEFTABLE
X'81E550'	DEFM	9	Binary(8)	Defaults table EDM. Indicates volume expiration is managed by an External Data Manager. 0 = NO 1 = YES FF = Blank	LC DEFTABLE
X'81E570'	DEFN	16	Character(variable length)	Defaults table program name mask. * = All program names match	LC DEFTABLE
X'81E600'	DEFO	9	Binary(8)	Defaults table retention override, indicates whether RETPD overrides the value specified in the JCL O = no 1 = yes	LC DEFTABLE
X'81E700'	DEFP	14	Character(variable length)	Defaults table scratch pool.	LC DEFTABLE
X'81E800'	DEFR	9	Binary(8)	Defaults table retention method default O = VRSEL 1 = EXPDT FF = blank	LC DEFTABLE
X'81E900'	DEFV	16	Character (variable length)	Defaults table VRS management value	LC DEFTABLE
X'81EA00'	DEFW	9	Binary(8)	Defaults table WHILECATALOG default 0 = OFF 1 = ON 2 = UntilExpired FF = Blank	LC DEFTABLE
X'81EB00'	DEFX	9	Binary(8)	Defaults table VRSEL Exclude default 0 = NO 1 = YES FF = Blank	LC DEFTABLE
X'81EC00'	DEFY	9	Binary(8)	Defaults table “Retain By” default 0 = Volume 1 = First file 2 = Volume set FF = Blank	LC DEFTABLE
X'81ED00'	DEFZ	9	Binary(8)	Defaults table continue, indicates whether we need to continue searching the defaults table to see if any other entries match. 0 = NO 1 = YES	LC DEFTABLE
X'81F000'	DEN	9	Binary(8)	Media density: 0=UNDEFINED 1=1600 2=6250 3=3480 4=COMPACT	LV, SV(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'820000'	DESC	38	Character (variable length)	Volume or VRS description	LS, LV, SS(e), SV(e)
X'821000'	DEST	16	Character (variable length)	Destination name	LV, SV
X'822000'	DEV	12	Character (fixed length)	Device number	LD, SD(e)
X'822500'	DKBC	12	Binary(32)	Data sets kept by catalog	LV, SV(e)
X'823000'	DLR/DLRJ	12	Packed decimal Julian date format	Date last referenced/read	LD, LV, LS, SD(e), SS, SV(e)
X'823700'	DLTD	9	Binary(8)	Deleted by disposition processing: 0=NO 1=YES	LD, SD(e)
X'824000'	DLWJ	12	Packed decimal Julian date format	Date last written	LD, LV, SD(e), SV(e)
X'825000'	DNM	52	Character (variable length)	Data set name mask	LC
X'825E00'	DPCT	9	Binary(8)	Percent of volume	LD, SD(e)
X'826000'	DPT	48	Character (variable length)	Owner's department	LO, SO
X'827000'	DRP	12	Binary(32)	Default retention period: Min 0, Max 93000 FFFFFFFE=PERMANENT	LC
X'828000'	DSC	12	Binary(32)	Data set count: Min 0, Max 4-digit	LV, SV
X'829000'	DSEQ	12	Binary(32)	Data set sequence: Min 0, Max 4-digit	LD, LV, SD(e), SV(e)
X'829500'	DSFX	10	Character (variable length)	Active defaults table suffix	LC OPT
X'82A000'	DSN	52	Character (variable length)	Data set name	LD, LV, SD, SV(e)
X'82A500'	DSPD	16	Character (variable length)	Disposition DD name	LC
X'82AA00'	DSPM	16	Character (variable length)	Disposition message prefix	LC
X'82B000'	DSR	9	Binary(8)	Data set recording: 0=NO 1=YES	LV, SV
X'82B030'	DSS6	22	Compound (Binary(8) Factor, Binary(64) Value)	Data set size, Factor: 0=bytes 1=KB 2=MB 3=GB 4=TB Value: Minimum value = 0.	LD, SD(e)
X'82B200'	DSTT	9	Binary(8)	Destination type 0=SHELF 1=STORE_BUILTIN 2=MANUAL 3=AUTO 4=STORE_BINS 5=STORE_NOBINS	LV
X'82B500'	DSXP	9	Binary(8)	DSNEXPIRE 0=NONE 1=BLOCK	LC
X'82BB00'	DSYS	16	Character (variable length)	Creating system ID	LV, SV(e)
X'82C000'	DTE	9	Binary(8)	Installation date format: 1=A 2=E 3=I 4=J	LC
X'82D000'	DTM	12	Packed decimal time format	Last store update run time	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'82D500'	EBIN	9	Binary(8)	Extended bin enable status 0=DISABLED 1=ENABLED	LC
X'82D700'	EDM	9	Binary(8)	0 EDM = N 1 EDM = Y	LC OPT, LV VOL, SV(e)
X'82DFF0'	EML	71	Character (variable length)	Owner's e-mail address, 1 to 63 characters	LO, SO
X'82E000'	EMN	16	Character (variable length)	Owner's node	LO, SO
X'82F000'	EMU	16	Character (variable length)	Owner's user ID	LO, SO
X'830000'	ETL	28	Character (variable length)	Telephone number	LO, SO
X'830800'	EXRB	9	Binary(8)	Retained By Volume = 0 Firstfile = 1 Set = 2	LC OPT, LV VOL, SV(e)
X'831000'	FCD	12	Character (variable length)	Feature code	LP, LV, SP, SV(e)
X'831800'	FCSP	9	Binary(8)	Catalog synchronize in progress: 0=NO 1=YES	LC
X'832000'	FDB	12	Binary(32)	Free bins in DISTANT location Min 0, Max 6-digit	LC
X'833000'	FILE	12	Binary(32)	Physical file sequence Min 1, Max 4-digit	LD, SD
X'834000'	FLB	12	Binary(32)	Free bin numbers in LOCAL location: Min 0, Max 6-digit	LC
X'835000'	FOR	28	Character (variable length)	Owner's forename	LO, SO
X'836000'	FRB	12	Binary(32)	Free bin numbers in REMOTE location: Min 0, Max 6-digit	LC
X'837000'	FRK	12	Binary(32)	Free rack numbers in library: Min 0, Max 10-digit	LC
X'837500'	FRXP	9	Binary(8)	FORCEEXPIRE 0=NO 1=YES	LD, SD(e)
X'837800'	GDGC	9	Binary(8)	GDG CYCLEBY: 0=Generation 1=Create order	LC
X'837805'	GDGD	9	Binary(8)	GDG DUPLICATE: 0=Bump from sub chain 1=Drop from chain 2=Keep 3=Count	LC
X'837900'	GDRP	12	Binary(32)	Default retention period for RM=EXPDT GDG data sets Min 0, Max 93000 FFFFFFFE=permanent	LC
X'837940'	GWCT	9	Binary(8)	Default WHILECATALOG setting for RM=EXPDT GDG data sets 0 = OFF 1 = ON 2 = UntilExpired	LC
X'838000'	GRK	14	Character (fixed length)	Generic rack number = reject prefix	LC
X'838F40'	HLD	9	Binary(8)	0=Hold No 1=Hold Yes	LV, SV(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'839000'	HLOC	16	Character (variable length)	Home location	LV, SV
X'839200'	HLOT	9	Binary(8)	Home location type 0=SHELF 1=STORE_BUILTIN 2=MANUAL 3=AUTO 4=STORE_BINS 5=STORE_NOBINS	LV
X'839600	HSKP	9	Binary(8)	HOUSEKEEPING 0=LIMITED 1=ALL	LC OPT
X'83A000'	INTR	9	Binary(8)	Volume intransit status: 0=NO 1=YES	LV, SV
X'83B000'	IPL	9	Binary(8)	Date check required on IPL: 0=NO 1=YES	LC
X'83B830'	IRMM	9	Binary(8)	Managed by IRMM, 0=NO 1=YES	LV, SV (e)
X'83C000'	ITL	16	Character (variable length)	Telephone number	LO, SO
X'83CA00'	JBDT	12	Packed decimal Julian date	Last Journal Backup Date	LC
X'83CB00'	JBTM	12	Packed decimal time format	Last Journal Backup Time	LC
X'83D000'	JDS	52	Character (variable length)	Journal name	LC
X'83E000'	JRNF	10	Binary(15)	JOURNALFULL parmlib value: 0 - 99	LC
X'83EA00'	JRNS	9	Binary(8)	Journal status: 0=Disabled 1=Enabled 2=Locked	LC
X'83ED00'	JRNT	9	Binary(8)	Journal transaction: 0=No 1=Yes	LC
X'83F000'	JRNU	10	Binary(15)	Journal percentage used: 0 - 100	LC
X'83F500'	KEL1	72	Character (variable length)	Key encryption key label 1	LV, SV(e)
X'83F505'	KEL2	72	Character (variable length)	Key encryption key label 2	LV, SV(e)
X'83F520'	KEM1	13	Character (variable length)	Key encoding mechanism for key label 1: LABEL or HASH	LV, SV(e)
X'83F525'	KEM2	13	Character (variable length)	Key encoding mechanism for key label 2: LABEL or HASH	LV, SV(e)
X'840000'	LBL	9	Bit(8)	Volume label type: '20'=NL '10'=AL '08'=SL '02'=BLP '01'=UL	LV, SV
X'841000'	LBN	12	Binary(32)	Bin numbers in LOCAL location Min 0, Max 6-digit	LC
X'841500'	LCDJ	12	Packed decimal Julian Date format	Last change date	LB, LD, LO, LP, LR, LV, LS, SD(e), SP(e), SS(e)
X'842000'	LCID	16	Character (variable length)	Last change user IDID starts with asterisk (*) for change made by DFSMSrmm	LB, LD, LO, LP, LR, LS, LV, SD(e), SP(e), SS(e), SV(e),
X'842500'	LCSI	16	Character (variable length)	Last change system ID	LB, LD, LO, LP, LR, LS, LV, SD(e), SP(e), SS(e), SV(e)
X'843000'	LCT	10	Binary(15)	Default lines per page Min 10, Max 3-digit	LC
X'843100'	LCTK	12	Binary (31)	Local tasks binary value	LC
X'843500'	LCTM	12	Packed decimal time format	Last change time	LB, LD, LO, LP, LR, LS, LV, SD(e), SP(e), SS(e), SV(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'843600'	LCUD	12	Packed decimal Julian Date format	Last user change date	LB, LD, LO, LP, LR, LS, LV, SD(e), SP(e), SS(e), SV(e)
X'843700'	LCUT	12	Packed decimal time format	Last user change time	LB, LD, LO, LP, LR, LS, LV, SD(e), SP(e), SS(e), SV(e)
X'844000'	LDDF	9	Binary(8)	Location definition exists: 0=NO 1=YES	LC
X'843B00'	LDD	16	Character (fixed length)	Last used DD name	LD, SD(e)
X'845000'	LDEV	12	Character (fixed length)	Last drive	LD, LV, SD(e), SV(e)
X'846000'	LDLC	16	Character (variable length)	Location name	LC
X'847000'	LDLT	9	Binary(8)	Location type: 0=SHELF 1=AUTO 2=MANUAL 3=STORE	LC
X'848000'	LDMN	16	Character (variable length)	Location media name	LC
X'849000'	LDMT	9	Binary(8)	Location management type: 0=UNDEFINED 1=BIN 2=NOBINS	LC
X'84A000'	LDPR	12	Binary(32)	Location priority: Min 0, Max 4-digit	LC
X'84A100'	LDAM	9	Binary(8)	Automove: 0= No 1= Yes	LC
X'84B000'	LINE	264	Character (variable length)	Output data line	All list and search subcommands
X'84B420'	LJOB	16	Character (fixed length)	Last used job name	LD, SD(e)
X'84C000'	LOAN	16	Character (fixed length)	Loan location	LV, SV
X'84D000'	LOC	16	Character (variable length)	Location	LB, LR, LS, LV, SB, SR, SS, SV
X'84E000'	LOCT	9	Binary(8)	Location type 0=SHELF 1=STORE_BUILTIN 2=MANUAL 3=AUTO 4=STORE_BINS 5=STORE_NOBINS 6=IN_CONTAINER	LV, SV(e)
X'84E760'	LPGM	16	Character (fixed length)	Last used program name	LD, SD(e)
X'84F000'	LRCL	12	Binary(32)	Logical record length: Min 0, Max 5-digit	LD, SD(e)
X'84F800'	LRED	12	Binary(32)	Last reference extra days Min 0, Max 93000	LC, LD, SD(e)
X'850000'	LRK	12	Binary(32)	Library rack numbers: Min 0, Max 10-digit	LC
X'850370'	LSTP	16	Character (variable length)	Last used step name	LD, SD(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'850500'	LVC	9	Binary(8)	Current label version: 0=No version specified 1=Label version 1 3=Label version 3 4=Label version 4	LV, SV
X'850A00'	LVN	9	Binary(8)	Required label version: 0=No version specified 3=Label version 3 4=Label version 4	LV, SV
X'851000'	MC	16	Character (variable length)	Management class	LD, SD(e)
X'851400'	MDNF	16	Character (8)	Media Information Name	LC, LV, SV(e)
X'851200'	MCAT	9	Binary(8)	SMS Management class attributes enabling 0 = NONE 1 = ALL 2 = VRSELXDI	LC
X'851980'	MDRA	12	Binary(32)	MEDINF replace policy for age	LC
X'8519C0'	MDRP	12	Binary(32)	MEDINF replace policy for permanent errors	LC
X'8519E0'	MDRT	12	Binary(32)	MEDINF replace policy for temporary errors	LC
X'8519F0'	MDRW	12	Binary(32)	MEDINF replace policy for write mount count	LC
X'851A00'	MDRX	16	Character (8)	External Recording Technology	LC, LV, SV(e)
X'852000'	MDS	52	Character (variable length)	Control data set name	LC
X'853000'	MDTJ	12	Packed decimal Julian date format	Control data set create date	LC
X'853400'	MDTX	16	Character (8)	External Media Type	LC, LV, SV(e)
X'854000'	MEDA	9	Binary(8)	Media special attributes: 0=NONE 1=RDCOMPAT	LV, SV
X'855000'	MEDC	9	Binary(8)	Media compaction 0=UNDEFINED 1=NO 2=YES	LV, SV
X'856000'	MEDN	16	Character (variable length)	Media name	CV, LC, LB, LR, LV, SB, SR, SV
X'857000'	MEDR	9	Binary(8)	Recording technology: 0=NON-CARTRIDGE 1=18TRK 2=36TRK 3=128TRK 4=256TRK 5=384TRK 6=EFMT1 7=EFMT2 8=EEFMT2 9=EFMT3 10=EEFMT3 11=EFMT4 12=EEFMT4	LV, SV, LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'858000'	MEDT	9	Binary(8)	Media type: 0=UNDEFINED 1=CST 2=ECCST 3=HPCT 4=EHPCT 5=ETC/MEDIA5 6=EWTC/MEDIA6 7=EETC/MEDIA7 8=EEWTC/MEDIA8 9=EXTC/MEDIA9 10=EXWTC/MEDIA10 11=EATC/MEDIA11 12=EAWTC/MEDIA12 13=EAETC/MEDIA13	LV, SV, LC
X'859000'	MFR	16	Character (variable length)	Source location name	LC
X'85A000'	MID	20	Character (variable length)	Mount message ID	LC
X'85A500'	MIV	14	Character (fixed length)	Moving-in volume	LB, SB
X'85A900'	MOV	14	Character (fixed length)	Moving-out volume	LB, SB
X'85B000'	MOVM	9	Binary(8)	Move mode: 0=AUTO 1=MANUAL	LV, SV(e)
X'85C000'	MOP	9	Binary(8)	Master overwrite: 1=ADD 2=LAST 3=MATCH 4=USER	LC
X'85D000'	MRP	12	Binary(32)	Maximum retention period: Min 0, Max 93000 -1 (negative) means unlimited retention.	LC
X'85E000'	MSGF	9	Binary(8)	Message text case: 0=MIXED 1=UPPER	LC
X'85F000'	MST	9	Binary(8)	Move status: 0=UNKNOWN 1=PENDING 2=CONFIRMED 3=COMPLETE	LC
X'860000'	MTM	12	Packed decimal time format	Control data set create time	LC
X'861000'	MTO	16	Character (variable length)	Target location name, installation defined name, SHELF, or SMS library name	LC
X'862000'	MTP	9	Binary(8)	Control data set type: 0=MASTER	LC
X'862800'	MTY	9	Binary(8)	Move type: 0=NOTRTS 1=RTS	LC
X'862B00'	MVBY	9	Binary(8)	Move by: 0=VOLUME 1=SET	LC
X'863000'	MVS	9	Binary(8)	MVS use 0=NO 1=YES	LV, SV(e)
X'864500'	NDRP	12	Binary(32)	Default retention period for RM=EXPDT non-GDG data sets Min 0, Max 93000 FFFFFFFE=permanent	LC
X'865000'	NLOC	16	Character (variable length)	Required location	LV, SV(e)
X'865200'	NLOT	9	Binary(8)	Required location type 0=SHELF 1=STORE_BUILTIN 2=MANUAL 3=AUTO 4=STORE_BINS 5=STORE_NOBINS	LV
X'866000'	NME	16	Character (variable length)	Security class name	LC, LD, LV, SD(e), SV(e)
X'866800'	NOT	9	Binary(8)	User notification: 0=NO 1=YES	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'867000'	NVL	14	Character (fixed length)	Next volume serial	LV, SV(e)
X'868000'	NVRS	16	Character (variable length)	Next VRS name	LS, SS
X'868500'	NWCT	9	Binary(8)	Default WHILECATALOG setting for RM=EXPDT non-GDG data sets 0 = OFF 1 = ON 2 = UntilExpired	LC
X'869000'	OAC	9	Binary(8)	Owner access 0=READ 1=UPDATE 2=ALTER	LV, SV(e)
X'86A000'	OBMN	16	Character (variable length)	Old bin number media name	LV, SV(e)
X'86B000'	OBN	14	Character (fixed length)	Old bin number	LV, SV(e)
X'86B800'	OCE	9	Binary(8)	Volume information recorded at O/C/EOV 0=NO 1=YES	LV, SV(e)
X'86C000'	OLOC	16	Character (variable length)	Old location	LV, SV(e)
X'86C100'	OLON	16	Character (variable length)	Old loan location	LV, SV(e)
X'86C200'	OLOT	9	Binary(8)	Old location type 0=SHELF 1=STORE_BUILTIN 2=MANUAL 3=AUTO 4=STORE_BINS 5=STORE_NOBINS 6=IN_CONTAINER	LV
X'86D000'	OPL	10	Binary(15)	Position of rack number or pool ID Min 1, Max 3-digit	LC Position in the message.
X'86E000'	OPM	9	Binary(8)	Operating mode 1=M 2=R 3=W 4=P	LC
X'86E8A0'	ORIA	9	Binary(8)	Input action: 0=ACCEPT 1=IGNORE 2=REJECT	LC
X'86E8A8'	ORII	9	Bit(8)	Input IGNORE condition (BY): X'80'=SPECIFIC X'40'=NONSPECIFIC X'C0'=ANY	LC
X'86E8B8'	ORIR	9	Bit(8)	Input REJECT condition (BY): X'80'=SYSID X'40'=CATLG	LC
X'86EA00'	OROA	9	Binary(8)	Output action: 0=ACCEPT 1=IGNORE 2=REJECT	LC
X'86EA08'	OROI	9	Bit(8)	Output IGNORE condition (BY): X'80'=SPECIFIC X'40'=NONSPECIFIC X'C0'=ANY	LC
X'86EA18'	OROR	9	Bit(8)	Output REJECT condition (BY): X'80'=SYSID X'40'=CATLG	LC
X'86EF08'	ORTP	9	Binary(8)	Type of open rule entry: 0=RMM 1=NORMM	LC
X'86EF80'	ORVS	14	Character (variable length)	Volume range start	LC
X'86EF85'	ORVL	14	Character (variable length)	Volume serial number, specific or generic	LC
X'86EF8F'	ORVE	14	Character (variable length)	Volume range end	LC
X'86F000'	OVL	10	Binary(15)	Position of volume serial number: Min 1, Max 3-digit	LC Position in the message.

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'86F500'	OVOL	14	Character (fixed length)	Old volume	LB, SB
X'870000'	OWN	16	Character (variable length)	Owner	GV, LD, LO, LP, LS, LV, SD(e), SO, SP, SS, SV
X'871000'	OXDJ	12	Packed decimal Julian date format	Original expiration date	LD, LV, SD(e), SV(e)
X'871800'	PACS	9	Binary(8)	PREACS 0=NO 1=YES	LC
X'871E00'	PDA	9	Binary(8)	PDA state: 0=Off 1=On 2=None	LC
X'871E10'	PDAC	9	Binary(8)	PDA block count: Numeric 2-255	LC
X'871E30'	PDAL	9	Binary(8)	PDA log state: 0=Off1=On	LC
X'871E90'	PDAS	9	Binary(8)	PDA block size: Numeric 1-31	LC
X'872000'	PDS	48	Character (variable length)	Pool description	LC
X'873000'	PDSC	40	Character (variable length)	Product description	LP, SP(e)
X'874000'	PEND	9	Bit(8)	Actions pending: '80'=SCRATCH '40'=REPLACE '20'=INIT '10'=ERASE '08'=RETURN '04'=NOTIFY	LV, SV
X'875000'	PID	14	Character (variable length)	Pool prefix	LC, LR, SR
X'876000'	PLN	16	Character (variable length)	Pool name	LC
X'877000'	PNME	38	Character (variable length)	Software product name	LP, SP
X'878000'	PNUM	16	Character (variable length)	Software product number	LP, LV, SP, SV(e)
X'879000'	PRD	12	Binary(32)	Permanent read errors: Min 0, Max 5-digit	LV, SV(e)
X'87A000'	PRF	9	Binary(8)	Pool definition RACF® (A component of the Security Server for z/OS) option: 0=NO 1=YES	LC
X'87B000'	PRTY	12	Binary(32)	Priority: Min 0, Max 4-digit	LS, SS
X'87C000'	PSFX	10	Character (fixed length)	Parmlib member suffix	LC
X'87C010'	PSF2	10	Character (fixed length)	Second parmlib member suffix	LC
X'87D000'	PSN	16	Character (variable length)	Pool definition system ID	LC
X'87D300'	PSZ6	22	Compound (Binary(8) Factor, Binary(64) Value)	Physical space used	LD, LV
X'87DB00'	PTNA	9	Binary(8)	NOSMT action for partition entry: 0=ACCEPT 1=IGNORE	LC
X'87DB0C'	PTNL	16	Character (variable length)	Location name	LC
X'87E000'	PTP	9	Binary(8)	Pool definition pool type: 0=SCRATCH 1=RACK	LC
X'87EB80'	PTSA	9	Binary(8)	SMT action for partition entry: 0=ACCEPT 1=IGNORE	LC
X'87EBA8'	PTTP	9	Binary(8)	Type of partition entry: 0=RMM 1=NORMM	LC
X'87EC00'	PTVS	14	Character (variable length)	Volume range start	LC
X'87EC08'	PTVL	14	Character (variable length)	Volume serial number, specific or generic	LC
X'87EC0F'	PTVE	14	Character (variable length)	Volume range end	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'87F000'	PVL	14	Character (fixed length)	Previous volume: 1 - 6 character	LV, SV(e)
X'880000'	PWT	12	Binary(32)	Permanent write errors: Min 0, Max 5-digit	LV, SV(e)
X'881000'	RBN	12	Binary(32)	Number of bin numbers in REMOTE location: Min 0, Max 6-digit	LC
X'881200'	RBYS	9	Binary(8)	Retain by set: 0=NO 1=YES	LV, SV(e)
X'882000'	RCF	9	Binary(8)	Installation RACF support: 1=N 2=P 3=A 4=C	LC
X'883000'	RCFM	12	Character (variable length)	RECFM	LD, SD(e)
X'884000'	RCK	14	Character (fixed length)	Rack or bin number	AB, AR, AV, CV, DB, DR, LB, LP, LR, LV, SB, SP(e), SR, SV
X'888500'	RLPR	12	Binary(32)	Required location priority	LV, SV(e)
X'886000'	RDTJ	12	Packed decimal Julian date format	Last control data set extract date	LC
X'888000'	RET	11	Binary(8)	Retention type: 1st byte: 1=RETAIN WHILE CATALOGED 2nd byte: 1=RETAIN UNTIL EXPIRED 3rd byte: 1=CYCLES 2=DAYS 3=REFDAYS 4=VOLUMES 5=EXTRA DAYS 6=BY DAYS CYCLE	LS, SS
X'888800'	RM	9	Binary(8)	Retention method: 0=VRSEL 1=EXPDT	LC OPT, LV VOL, SV(e)
X'889000'	RMID	25	Character (variable length)	Started procedure name. Up to 17 characters. One of: • procedure name • job name • concatenation of procedure name.identifier	LC
X'888A00'	RMSB	9	binary(8)	Retention method set by 0=blank (not set) 1=CMD 2=CMD_DEF 3=OCE_DEF 4=OCE_EXIT 5=LCS_DEF 6=CNVT 7=EXPORT_DEF 8=INERS_DEF 9=MC_ATTR 10=DEFTABLE	LV VOL, SV(e)
X'88A000'	RST	9	Binary(8)	Rack or bin status 0=EMPTY 1=FREE 2=INUSE	LB, LR, SB, SR
X'88B900'	RTBY	9	Binary(8)	Retain by: 0=VOLUME 1=SET	LC
X'88C000'	RTDJ	12	Packed decimal Julian date format	Retention date	LD, LV, SD, SV

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'88E000'	RTM	12	Packed decimal time format	Last control data set extract time	LC
X'88E500'	RUB	9	Binary(8)	Reuse bin at 0=CONFIRMMOVE 1=STARTMOVE	LC
X'890000'	SC	16	Character (variable length)	Storage class name	LD, SD(e)
X'891000'	SCRM	9	Binary(8)	Binary value 0=Auto 1=manual	LC
X'892000'	SCST	9	Bit(8)	Security class status '80'=SMF '40'=MSGOPT '20'=ERASE	LC
X'894000'	SC1	12	Binary(32)	Storenumber Min 1, Max 5-digit	LS, SS, SS(e)
X'895000'	SDTJ	12	Packed decimal Julian date format	Movement tracking date	LV, SV(e)
X'896000'	SEC	9	Binary(8)	Security class number Min 0, Max 255	LC
X'898000'	SEQ	12	Binary(32)	Volume sequence Min 1, Max 9999	LV, SV
X'89A000'	SG	16	Character (variable length)	Storage group name	LD, LV, SD(e), SV(e)
X'89B000'	SID	16	Character (variable length)	DFSMSrmm system ID	LC
X'89C000'	SLM	10	Binary(15)	MAXHOLD value Min 10, Max 500	LC
X'89E000'	SMI'	10	Binary(15)	Offset to message ID Min 0, Max 3-digit	LC
X'89E210'	SMP	9	Binary(8)	System-managed tape purge: 0=NO 1=YES 2=ASIS	LC
X'89E220'	SMU	9	Bit(8)	System-managed tape update: 20=Command 40=Scratch 80=Exits N/A	LC
X'89F000'	SOSJ	12	Packed decimal Julian date format	Last expiration processing start date	LC
X'8A0000'	SOSP	16	Character (variable length)	Scratch procedure name	LC
X'8A1000'	SOST	12	Packed decimal time format	Last expiration processing start time	LC
X'8A1A00'	SRHN	71	Character (variable length)	Server host name 1-to-63 alphanumeric characters including hyphen, period, and blank	LC
X'8A1A30'	SRIP	53	Character (variable length)	Server IP address 1-to-45 numeric characters including colon, period, and blank	LC
X'8A1A50'	SRPN	12	Binary (31)	Server number binary value	LC
X'8A1AF0'	SRTK	12	Binary (31)	Server tasks binary value	LC
X'8A2000'	SSM	10	Binary(15)	SMF security record type: 128-255, 42, or 0	LC
X'8A2500'	SSTY	9	Binary (8)	Subsystem type 0=Standard system 1=Client system 2=Server system	LC
X'8A2800'	STDS	9	Bit (8)	Debug setting X'80' OCE X'40' SNAP	LC
X'8A3000'	STEP	16	Character (variable length)	Step name	LD, SD(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8A3200'	STIS	9	Binary (8)	Task - IP verb state 0=NONE 1=STARTED 2=ENDED	LC
X'8A3201'	STIT	12	Packed decimal time format	Task - IP verb time	LC
X'8A3203'	STIV	9	Binary (8)	Task - IP verb 0=NONE 1=READ 2=WRITE 3=CONNECT 4=CLOSE	LC
X'8A3300'	STLA	10	Binary (15)	Local active tasks Numeric: 0-999	LC
X'8A3307'	STLH	10	Binary (15)	Local held tasks Numeric: 0-999	LC
X'8A3314'	STLO	10	Binary (15)	Local tasks Numeric: 0-999	LC
X'8A3317'	STLR	12	Packed decimal time format	Last RESERVE time	LC
X'8A3400'	STNH	9	Binary (8)	New requests held 0=NOTHELD 1=HELD	LC
X'8A3450'	STPL	9	Bit (8)	PDA trace levels X'80' level 1 trace X'40' level 2 trace X'20' level 3 trace X'10' level 4 trace	LC
X'8A3500'	STQC	12	Binary (32)	Catalog requests Numeric: 0-999999	LC
X'8A3511'	STQN	12	Binary (32)	Nowait requests Numeric: 0-999999	LC
X'8A3515'	STQR	12	Binary (32)	Queued requests Numeric: 0-999999	LC
X'8A3600'	STRF	13	Character (variable length)	Task - requested function	LC
X'8A3602'	STRH	9	Binary (8)	CDS RESERVEd 0=DEQ 1=ENQ	LC
X'8A3607'	STRM	9	Binary (8)	RMM status: 0=ACTIVE 1=RESET 2=QUIESCED	LC
X'8A3614'	STRT	16	Character (variable length)	Task - requestor's system	LC
X'8A3650'	STSA	10	Binary (15)	Server active tasks Numeric: 0-999	LC
X'8A3657'	STSH	10	Binary (15)	Server held tasks Numeric: 0-999	LC
X'8A3661'	STSL	9	Binary (8)	Server listener task status 0=Standard or client system 1=task is active 2=task not active	LC
X'8A3664'	STSO	10	Binary (15)	Server tasks Numeric: 0-999	LC
X'8A3669'	STST	12	Packed decimal time format	Task - Start time	LC
X'8A3700'	STTQ	16	Character (variable length)	Task - requestor	LC
X'8A3701'	STTR	11	Character (variable length)	Task - requestor's type: JOB, STC, TSU.	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8A3702'	STTS	9	Binary (8)	Task - status 0=NONE 1=HOLD 2=CANCEL 3=RESERVE	LC
X'8A3703'	STTT	12	Binary (32)	Task - Token Hexadecimal value: X'00000000' - X'FFFFFFFF'	LC
X'8A3800'	STVC	12	Binary(32)	Count of volumes stacked on a stacked volume	LV VOL, SV(e)
X'8A4000'	SUR	28	Character (variable length)	Surname	LO, SO
X'8A5000'	SYS	16	Character (variable length)	SMF System ID	LD, LV, SD(e), SV(e)
X'8A6000'	TAC	9	Binary(8)	Reject type 0=ANYUSE 1=OUTPUT	LC
X'8A6800'	TLR	12	Packed decimal time format	hhmmsstC, where hhmmsst is the time in hours, minutes, seconds, and tenths of seconds and C is a standard packed-decimal sign character.	LS, SS
X'8A7000'	TRD	12	Binary(32)	Temporary read errors Min 0, Max 5-digit	LV, SV(e)
X'8A7800'	TVXD	12	Binary(32)	TVEXTPURGE days	LC OPT
X'8A7900'	TVXP	9	Binary(8)	Tape volume exit purge option: 0=RELEASE 1=EXPIRE 2=NONE	LC
X'8A8000'	TWT	12	Binary(32)	Temporary write errors: Min 0, Max 5-digit	LV, SV(e)
X'8A9000'	TYP	9	Bit(8)	VRS type: '80'=GDG '40'=PSEUDGDG '20'=DSNAME '10'=VOLUME '08'=NAME	LS, SS
X'8A9E00'	TZ	12	Binary(32)	Signed number; the offset from common time in seconds. When non-zero, use this value to adjust all dates and times from the DFSMSrmm systems' local time to common time.	All
X'8AA000'	UDTJ	12	Packed decimal Julian date format	Late update date	LC
X'8AB001'	UID	16	Character (variable length)	User ID. The SFI is incremented by one for each UID that is found. (X'8AB001'-X'8AB00C')	LV, SV(e)
X'8AC000'	UNC	9	Binary(8)	Uncatalog option: 0=N 1=Y 2=S	LC
X'8AD000'	USEC	12	Binary(32)	Volume use count: Min 0, Max 5-digit	LV, SV(e)
X'8AE000'	USEM	12	Binary(32) unsigned	Volume usage (KB): Min 0, Max 4294967295. 4294967295 indicates that USE6 must be used.	LV, SV(e)
X'8AE030'	USE6	22	Compound (Binary(8) Factor, Binary(64) Value)	Volume usage, Factor: 0=bytes 1=KB 2=MB 3=GB 4=TB Value: Minimum value = 0.	LV, SV(e)
X'8AE600'	UTC	9	Binary(8)	Common Time: 0=DISABLED 1=ENABLED	LC

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8AE800'	UTM	12	Packed decimal time format	Late update time	LC
X'8AF001'	VAC	9	Binary(8)	Volume access: 0=NONE 1=READ 2=UPDATE	LV, SV(e)
X'8B0000'	VACT	9	Binary(8)	VRSMIN action: 0=FAIL 1=INFO 2=WARN 3=OFF	LC
X'8B0800'	VANX	9'	Binary(8)	Next VRS type: 0=Undefined 1=Next 2=And	LS, SS
X'8B0B00'	VCAP	12	Binary(32)	Volume/Media capacity	LC, LV, SV(e)
X'8B1000'	VCHG	9	Binary(8)	VRSCHANGE value: 0=INFO 1=VERIFY	LC
X'8B2000'	VDD	10	Binary(15)	VRS delay days: Min 0, Max 99	LS, SS(e)
X'8B2800'	VDRA	9	Binary(8)	VRSDROP action: 0=FAIL 1=INFO 2=WARN 3=OFF	LC
X'8B2802'	VDRC	12	Binary(32)	VRSDROP count	LC
X'8B280F'	VDRP	10	Binary(15)	VRSDROP percent	LC
X'8B3000'	VDTJ	12	Packed decimal time format	Last inventory management processing date	LC
X'8B4000'	VER	14	Character (variable length)	Software produce version, release, modification **vvrrmm**	LP, LV, SP, SV(e)
X'8B4100'	VEX	9	Binary(8)	VRSEL exclude: 0=No 1=Yes	LD, SD(e)
X'8B5000'	VJBN	16	Character (variable length)	Primary VRS job name	LD, LS, SD(e), SS
X'8B6000'	VLN	12	Binary(32)	Number of volumes: Min 0, Max 3-digit	LO, LP, SO, SP
X'8B7000'	VM	9	Binary(8)	VM use: 0=NO 1=YES	LV, SV(e)
X'8B8000'	VMIN	12	Binary(32)	VRSMIN count value: Min 0, Max 6-digit	LC
X'8B9000'	VMV	16	Character (variable length)	VRS management value	LD, SD(e)
X'8B9100'	VWMC	12	Binary(32)	Volume write mount count	LV, SV(e)
X'8B9E00'	VNDR	16	Character (8)	Vendor information	LV, SV(e)
X'8BA000'	VNME	52	Character (variable length)	Primary VRS name	LD, SD(e)
X'8BC000'	VOL	14	Character (fixed length)	1 - 6 characters volume serial	AV, CV, GV, LB, LD, LP, LR, LV, SB, SD, SP, SR, SV
X'8BC200'	VOLT	9	Binary(8)	Volume type: 0=PHYSICAL 1=LOGICAL 2=STACKED	LV, SV(e)
X'8BCD00'	VOL1	14	Character (fixed length)	VOL1 label volume serial number	LV, SV(e)
X'8BC300'	VPCT	9	Binary(8)	Volume percent full	LV, SV(e)
X'8BD000'	VRC	12	Binary(32)	Vital record count: Min 1, Max 5-digit	LS, SS, SS(e)

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8BD500'	VREA	9	Binary(8)	VRSRETAIN action: 0=FAIL 1=INFO 2=WARN 3=OFF	LC
X'8BD502'	VREC	12	Binary(32)	VRSRETAIN count	LC
X'8BD50F'	VREP	10	Binary(15)	VRSRETAIN percent	LC
X'8BE000'	VRJ	9	Binary(8)	VRS job name: 1 or 2	LC
X'8BF000'	VRS	52	Character (variable length)	Vital record specification name	LS, SS
X'8BF500'	VRSI	9	Binary(8)	Release action scratch immediate: 0=NO 1=YES	LS, LV, SS, SV(e)
X'8BFA00'	VRSL	9	Binary(8)	VRSEL value: 1=NEW	LC
X'8C0000'	VRSR	9	Binary(8)	VRS retained status: 0=NO 1=YES	LD, SD, SD(e)
X'8C0800'	VRXI	9	Binary(8)	Expiration date ignore: 0=NO 1=YES	LV, LS, SS, SV(e)
X'8C1000'	VSCD	12	Packed decimal Julian date format	Primary VRS subchain start date	LD, SD(e)
X'8C1800'	VSCN	16	Character (variable length)	Primary VRS subchain name	LD, SD(e)
X'8C2000'	VST	9	Bit(8)	Volume status: '80'=MASTER '40'=SCRATCH '20'=USER '10'=INIT '08'=ENTRY	LV, SV
X'8C3000'	VTM	12	Packed decimal time format	Last inventory management VRS time	LC
X'8C4000'	VTYP	9	Binary(8)	Matching VRS type: 0=UNDEFINED 1=DATASET 2=SMSMC 3=VRSMV 4=DSNMV 5=DSNMC	LD, SD(e)
X'8C4150'	WCTL	9	Binary(8)	WHILECATALOG setting for data set 0 = OFF 1 = ON 2 = UntilExpired	LD, SD(e)
X'8C4300'	WORM	9	Binary(8)	Volume is WORM: 0=NO 1=YES	LV, SV (e)
X'8C4500'	WWID	32	Character (24)	World-wide identifier	LV, SV (e)
X'8C5000'	XDC	9	Binary(8)	Expiration date check: 0=NO 1=YES 2=OPERATOR	LC
X'8C5D00'	XDRA	9	Binary(8)	EXPDTDROP action: 0=FAIL 1=INFO 2=WARN 3=OFF	LC
X'8C5D02'	XDRC	12	Binary(32)	EXPDTDROP count	LC
X'8C5D0F'	XDRP	10	Binary(15)	EXPDTDROP percent	LC
X'8C6000'	XDTJ	12	Packed decimal Julian date format	Expiration date	LC, LD, LV, SD, SV

Table 16. Command structured field introducers (continued)

SFI Number	SFI Name	SFI Length	SFI Data Type	Data Description	Subcommand
X'8C6100'	XDSB	9	Binary(8)	Expiration date set by 0=blank (not set) 1=CMD 2=CMD_DEF 3=CMD_VOLCAT 4=OCE_JFCB 5=OCE_EXIT 6=OCE_DEF 7=OCE_MAX 8=OCE_VOLCAT 9=LCS 10=LCS_DEF 11=TVEXTPURGE 12=CNVT 13=EXPORT 14=LASTREF 15=OCE_MC 16=CATRETPD 17=CATLG_DAYS 18=DEFTABLE	LD, LV VOL, SD(e), SV(e)
X'8C7000'	XTM	12	Packed decimal time format	Last inventory management expiration time	LC
				Expiration time	LD, LV, SD(e), SV(e)
X'8C7800'	X100	9	Binary(8)	EDG_EXIT100 installation exit status: **0** Exit is not defined or no exit modules exist **1** At least one active exit module exists **2** One or more exit modules exist, but none is active	LC
X'8C7801'	X200	9	Binary(8)	EDG_EXIT200 installation exit status: **0** Exit is not defined or no exit modules exist **1** At least one active exit module exists **2** One or more exit modules exist, but none is active	LC
X'8C7802'	X300	9	Binary (8)	EDG_EXIT300 installation exit status: **0** Exit is not defined or no exit modules exist **1** At least one active exit module exists **2** One or more exit modules exist, but none is active	LC
X'8C8000'	2JBN	16	Character (variable length)	Secondary VRS jobname mask	LD, SD(e)
X'8C9000'	2NME	16	Character (variable length)	Secondary VRS mask	LD, SD(e)
X'8CA000'	2SCD	12	Packed decimal Julian date format	Secondary VRS subchain start date	LD, SD(e)
X'8CB000'	2SCN	16	Character (variable length)	Secondary VRS subchain name	LD, SD(e)

Appendix B. Structured field introducers by subcommand

Table 17 on page 87 lists the structured field introducers by DFSMSrmm TSO subcommand.

The RMM SEARCHDATASET, RMM SEARCHPRODUCT, RMM SEARCHVOLUME, and RMM SEARCHVRS subcommands return different sets of structured field introducers depending on whether you specify the EDGXCI macro EXPAND=YES or EXPAND=NO parameter. When you specify the EXPAND=YES parameter, these subcommands return the same information as their corresponding RMM LIST subcommands: RMM LISTDATASET, RMM LISTPRODUCT, RMM LISTVOLUME, and RMM LISTVRS.

Table 17. Structured field introducers by subcommand

Subcommand	Structured field introducers
ADDBIN	CNT ENTN MSGL MSGN RCK RSNC RTNC SVCN
ADDDATASET	ENTN MSGL MSGN RSNC RTNC SVCN
ADDOWNER	ENTN MSGL MSGN RSNC RTNC SVCN
ADDPRODUCT	ENTN MSGL MSGN RSNC RTNC SVCN
ADDRACK	CNT ENTN MSGL MSGN RCK RSNC RTNC SVCN
ADDVOLUME	CLIB CNT CSG ENTN FRC FRS MSGL MSGN RCK RSNC RTNC SVCN VOL
ADDVRS	ENTN MSGL MSGN RSNC RTNC SVCN
CHANGEDATASET	ENTN MSGL MSGN RSNC RTNC SVCN
CHANGEOWNER	ENTN MSGL MSGN RSNC RTNC SVCN
CHANGEPRODUCT	ENTN MSGL MSGN RSNC RTNC SVCN
CHANGEVOLUME	CLIB CSG ENTN FRC FRS MEDN MSGL MSGN RCK RSNC RTNC SVCN
CHANGEVRS	ENTN MSGL MSGN RSNC RTNC SVCN
DELETEBIN	CNT ENTN MSGL MSGN RCK RSNC RTNC SVCN
DELETEDATASET	ENTN MSGL MSGN RSNC RTNC SVCN
DELETEOWNER	ENTN MSGL MSGN RSNC RTNC SVCN
DELETEPRODUCT	ENTN MSGL MSGN RSNC RTNC SVCN
DELETERACK	CNT ENTN MSGL MSGN RCK RSNC RTNC SVCN
DELETEVOLUME	CLIB ENTN FRC FRS MSGL MSGN RSNC RTNC SVCN
DELETEVRS	ENTN MSGL MSGN RSNC RTNC SVCN
GETVOLUME	ENTN FRC FRS MSGL MSGN OWN RSNC RTNC SVCN VOL
LISTBIN	ENTN LCDJ LCID LCSI LCTM LCUD LCUT LINE LOC MIV MOV MEDN MSGL MSGN OVOL RCK RSNC RST RTNC SVCN TZ VOL
LISTCONTROL ACTIONS	ACT AST RC

Table 17. Structured field introducers by subcommand (continued)

Subcommand	Structured field introducers
LISTCONTROL CNTL	ACS AUD BDT BTM CDSQ CDSU CSHN CSIP CSVE DBN CDS CDSF CSDT CSTM CTLD DDT DRP DTE DTM EBIN FBP FCSP FDB FEP FKP FLB FRB FRK FRP FSP FTP FVP FXP GDRP GWCT IPL JBDT JBTM JDS JRNS JRNU LBN LCT LRK MDS MDT MRP MTM MTP NDRP NOT NWCT OPM PACS RBN RC RCF RDT RMID RTM RUB SAT SDT SID SLM SOSD SOSP SOST SSM STM UDT UTC UTM VDT VTM XDTJ XTM X100 X200 X300
LISTCONTROL DEFTABLE	DEFD DEFE DEFJ DEFK DEFL DEFM DEFN DEFO DEFP DEFR DEFV DEFW DEFX DEFY DEFZ
LISTCONTROL LOCDEF	LDAM LDDF LDLC LDLT LDMN LDMT LDPR RC
LISTCONTROL MNTMSG	MID OPL OVL RC SMI
LISTCONTROL MEDINF	MDNF MDRA MDRP MDRT MDRW MDRX MDTX MEDR MEDT VCAP
LISTCONTROL MOVES	MFR MST MTO MTY
LISTCONTROL OPENRULE	ORIA ORII ORIR OROA OROI OROR ORTP ORVE ORVL ORVS
LISTCONTROL OPTION	ACCT AUD BLP BKPP CATS CDS CMDD CMDO CRP DEFC DRP DSFX DSPD DSPM DSXP DTE EDM EXRB GDGC GDGD HSKP IPL JDS JRNF JRNT LCT LCTK LRED MCAT MDS MEDN MOP MRP MSGF MVBY OPM NOT PDAC PDA PDAC PDAL PDAS PSFX PSF2 RC RCF RM RTBY RUB SID SLM SMP SMUC SMUE SMUS SOSP SRHN SRIP SRPN SRTK SSM SSTY TVXD TVXP UNC VACT VCHG VDRA VDRC VDRP VMIN VREA VREC VREP VRJ VRSL XDRA XDRC XDRP
LISTCONTROL PRTITION	PTNA PTNL PTSA PTTP PTVE PTVL PTVS
LISTCONTROL REJECT	GRK RC TAC
LISTCONTROL SECCLS	CLS ERS MSG NME RC SEC SMF
LISTCONTROL SECLEVEL	CLS DNM ERS MSG NME RC SEC SMF
LISTCONTROL STATUS	STDS STIS STIT STIV STLA STLH STLO STLR STNH STPL STQC STQN STQR STRF STRH STRM STRT STSA STSH STSL STSO STST STTQ STTR STTS STTT
LISTCONTROL VLPOOL	ACT MEDN MOP PDS PID PLN PRF PSN PTP SCRM XDC
LISTDATASET	ABND BESK BLKC BLKS BLKT BLK6 CDTJ CJBN CLS CPGM CRAT CTLG CTM CTRT DC DD DEV DLRJ DLTD DLWJ DPCT DSEQ DSN DSS6 ENTN FILE FRXP LCDJ LCID LCSI LCTM LCUD LCUT LDD LDEV LINE LPGM LRCL LRED LSTP MC MSGL MSGN NME OWN OXDJ PSZ6 RCFM RSNC RTDJ RTNC SC SG STEP SVCN SYS TZ VEX VJBN VNME VOL VRSR VSCD VSCN VTYP WCTL XDSB XDTJ XTM 2JBN 2NME 2SCD 2SCN
LISTOWNER	ADL DPT EML EMN EMU ENTN ETL FOR ITL LCDJ LCID LCSI LCTM LCUD LCUT LINE MSGL MSGN OWN RSNC RTNC SUR SVCN TZ VLN
LISTPRODUCT	ENTN FCD LCDJ LCID LCSI LCTM LCUD LCUT LINE MSGL MSGN OWN PDSC PNME PNUM RCK RSNC RTNC SVCN TZ VER VLN VOL
LISTRACK	ENTN LCDJ LCID LCSI LCTM LCUD LCUT LINE LOC MEDN MSGL MSGN PID RCK RSNC RST RTNC SVCN VOL

Table 17. Structured field introducers by subcommand (continued)

Subcommand	Structured field introducers
LISTVOLUME	ACN ACT ADTJ ATM AVL BIN BMN CDTJ CJBN CLS CRAT CRID CTM CTNR CTRT DBIN DBMN DEN DESC DEST DKBC DLRJ DLWJ DSC DSEQ DSN DSR DSTT D12 EDM ENTN EXRB FCD HLD HLOC HLOT INTR KEL1 KEL2 KEM1 KEM2 LBL LCDJ LCID LCSI LCTM LCUD LCUT LDEV LINE LOAN LOC LOCT LVC LVN MDNF MDRX MDTX MEDA MEDC MEDN MEDR MEDT MOVM MSGL MSGN MVS NLOC NLOT NME NVL OAC OBMN OBN OCE OLOC OLON OLOT OWN OXDJ PEND PNUM PRD PSZ6 PVL PWT RBYS RCK RLPR RM RMSB RSNC RTDJ RTNC SDTJ SEQ SG STVC SVCN TRD TWT TZ UID01 UID02 UID03 UID04 UID05 UID06 UID07 UID08 UID09 UID10 UID11 UID12 USEC USEM USE6 VAC VCAP VER VM VMIN VNDR VOL VOLT VOL1 VPCT VRSI VRXI VST VWMC WORM WWID XDSB XDTJ XTM
LISTVRS	DDTJ DESC DLRJ ENTN LCDJ LCID LCSI LCTM LCUD LCUT LINE LOC MSGL MSGN NVRS OWN PRTY RET RSNC RTNC SC1 SVCN TLR TYP TZ VANX VDD VJBN VRC VRS VRSI VRXI
SEARCHBIN	CONT ENTN LINE LOC MEDN MIV MOV MSGL MSGN OVOL RCK RSNC RST RTNC SVCN TZ VOL
SEARCHDATASET	CDTJ CONT CTM DSN ENTN FILE KEYF KEYT LINE LRED MSGL MSGN OWN OXDJ RSNC RTDJ RTNC SVCN VOL XDTJ
SEARCHDATASET(EXPAND=YES)	The same SFIs as the LISTDATASET subcommand.
SEARCHOWNER	ADL CONT DPT EML EMN EMU ETL FOR ITL OWN SUR TZ VLN
SEARCHPRODUCT	CONT ENTN FCD LINE MSGL MSGN OWN PNME PNUM RSNC RTNC SVCN VER VLN VOL
SEARCHPRODUCT(EXPAND=YES)	The same SFIs as the LISTPRODUCT subcommand.
SEARCHRACK	CONT ENTN LINE LOC MEDN MSGL MSGN PID RCK RSNC RST RTNC SVCN VOL
SEARCHVOLUME	ADTJ AVL CONT DESC DSC DSR ENTN EXRB HLD HLOC INTR KEYF KEYT LBL LINE LOAN LOC LVC LVN MDNF MDRX MDTX MEDA MEDC MEDN MEDR MEDT MSGL MSGN OWN PEND RCK RSNC RTDJ RTNC SEQ SVCN TYPF TYPT VCAP VOL VST XDTJ
SEARCHVOLUME(EXPAND=YES)	The same SFIs as the LISTVOLUME subcommand.
SEARCHVRS	CONT DDTJ ENTN LINE LOC MSGL MSGN NVRS OWN PRTY RET RSNC RTNC SVCN VANX VJBN VRS VRSI VRXI
SEARCHVRS(EXPAND=YES)	The same SFIs as the LISTVRS subcommand.

Appendix C. DFSMSrmm application programming interface mapping macros

DFSMSrmm API macros can be used to generate mappings: This section discusses:

- The parameter list generated by the list form of the EDGXCI macro, as shown in "EDGXCI: Parameter list" on page 91
- The structured field definitions generated by the EDGXSF macro, as shown in "EDGXSF: Structured field definitions" on page 91

EDGXCI: Parameter list

The mapping of the parameter list is generated by the list form of the EDGXCI macro.

The EDGXCI mapping macro is provided for information only. Although the fields and values of the parameter list are shown here, your application program should not directly access and modify the parameter list. Always use macro EDGXCI.

```
MYPL     DS    0D                ++ EDGXCI PARM LIST
MYPL_XVERSION DS XL1             ++ INPUT XVERSION
MYPL_XOPERATION DS XL1           ++  XOPERATION
MYPL_XOPERATION_BEGIN EQU 0      ++ XOPERATION.BEGIN KEYWORD
MYPL_XOPERATION_CONTINUE EQU 1   ++ XOPERATION.CONTINUE KEYWORD
MYPL_XOPERATION_RELEASE EQU 2    ++ XOPERATION.RELEASE KEYWORD
MYPL_XOPERATION_ENDALL EQU 3     ++ XOPERATION.ENDALL KEYWORD
MYPL_XOUTPUT DS XL1              ++  XOUTPUT
MYPL_XOUTPUT_LINES EQU 0         ++ XOUTPUT.LINES KEYWORD
MYPL_XOUTPUT_FIELDS EQU 1        ++ XOUTPUT.FIELDS KEYWORD
MYPL_XEXPAND DS XL1              ++  XEXPAND
MYPL_XEXPAND_YES EQU 0           ++ XEXPAND.YES KEYWORD
MYPL_XEXPAND_NO EQU 1            ++ XEXPAND.NO KEYWORD
MYPL_XAPIADDR DS A               ++   XAPIADDR
MYPL_XOUTBUFADDR DS A            ++   XOUTBUFADDR
MYPL_XSUBCMDADDR DS A            ++   XSUBCMDADDR
MYPL_XTOKEN DS CL4               ++   XTOKEN
MYPL_XMULTI DS XL1               ++  XMULTI
MYPL_XMULTI_NO EQU 0             ++ XMULTI.NO KEYWORD
MYPL_XMULTI_YES EQU 1            ++ XMULTI.YES KEYWORD
MYPL_XRSV0001 DS CL7             ++ RESERVED  XRSV0001
MYPL_XRSV0002 DS CL4             ++ RESERVED  XRSV0002
MYPL_XRSV0003 DS CL8             ++ RESERVED  XRSV0003
MYPLL    EQU   *-MYPL            ++ LENGTH OF PLIST
```

Figure 35. Mapping of the parameter list using the list form of EDGXCI

EDGXSF: Structured field definitions

Use macro EDGXSF in your application program to define the data that the DFSMSrmm API returns in your output buffer. This section includes:

- "EDGXSF parameters" on page 91
- "EDGXSF mapping" on page 92
- "EDGXSF labeling conventions" on page 150

EDGXSF parameters

The EDGXSF parameters are:

DSECT=YES
DSECT=NO
An optional parameter that specifies whether a DSECT statement is generated. The default is DSECT=YES.

DSECT=YES
Indicates that a DSECT statement should be generated.

DSECT=NO
Indicates that a DSECT statement should not be generated.

,LIST=YES
,LIST=NO
An optional parameter that specifies whether the macro expansion is printed. The default is LIST=YES.

,LIST=YES
Indicates to print the expansion.

,LIST=NO
Indicates do not print the expansion.

,TITLE=YES
,TITLE=NO
An optional parameter that specifies whether the macro title is printed. The default is TITLE=YES.

,TITLE=YES
Indicates to print the title.

,TITLE=NO
Indicates do not print the title

EDGXSF mapping

Always use macro EDGXSF to determine the exact labels used to define the DFSMSrmm structured field introducers. The tables in this topic show the dummy control section and the data types that define the generic mapping for the structured field introducers defined in Appendix A, “Structured field introducers (SFIs),” on page 63.

```
Common name: API Structure Field Introducers
Macro ID: EDGXSF
DSECT name: XSF_SFI
Owning component: DFSMSrmm (DF186)
Eye-catcher ID: None
Storage attributes: Subpool: user specified
                    Key: any key
                    Residency: 31 bit
Size: Variable
Created by: Caller
Pointed to by: N/A
Serialization: None
Function: The XSF_SFI area is initialized by
                    DFSMSrmm when an API call is made
                    via the EDGXCI executable macro
```

Table 18. Structure XSF_OUTBUF

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	*	XSF_OUTBUF	Output buffer
0	(0)	SIGNED	4	XSF_OUTBUF_BUFLNG	Output buffer length
4	(4)	SIGNED	4	XSF_OUTBUF_RQDLNG	Required buffer length
8	(8)	SIGNED	4	XSF_OUTBUF_DATALNG	Length of output data
12	(C)	CHARACTER	*	XSF_OUTBUF_FIELDS	Start of structured fields

```
   Structured Field Introducers for Structured Fields
```

Table 19. Structure XSF_SFI

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	*	XSF_SFI	Structured field introducers
0	(0)	CHARACTER	8	XSF_SFI_HD	
0	(0)	SIGNED	2	XSF_SFI_LENGTH	Length
2	(2)	CHARACTER	3	XSF_SFI_ID	Identifier
2	(2)	CHARACTER	2	XSF_SFI_IDVAL	Identifier value
4	(4)	CHARACTER	1	XSF_SFI_IDQUAL	Identifier qualifier
5	(5)	UNSIGNED	1	XSF_SFI_TYPE	Type
7	(7)	UNSIGNED	1	XSF_SFI_DTYPE	Data type
8	(8)	CHARACTER	*	XSF_SFI_DATA	Start of data

Compound SFI definition

Table 20. Structure XSF_SFI_COMPTYPE1

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
8	(8)	STRUCTURE	14	XSF_SFI_COMPTYPE1	Compound section
8	(8)	CHARACTER	6	XSF_SFI_COMPDATA	
8	(8)	CHARACTER	6	XSF_SFI_COMPHDR	Compound header
8	(8)	CHARACTER	6	XSF_SFI_COMPENT	Compound entry
8	(8)	UNSIGNED	1	XSF_SFI_COMPTYPE	Compound type
9	(9)	CHARACTER	3	XSF_SFI_FIELD1	
9	(9)	UNSIGNED	1	XSF_SFI_LEN1	Length of first field
10	(A)	UNSIGNED	1	XSF_SFI_DTYP1	Type of first field
11	(B)	UNSIGNED	1	XSF_SFI_FACTOR	Factor for second field
12	(C)	CHARACTER	2	XSF_SFI_FIELD2	
12	(C)	UNSIGNED	1	XSF_SFI_LEN2	Length of second field
13	(D)	UNSIGNED	1	XSF_SFI_DTYP2	Type of second field
14	(E)	CHARACTER	8	XSF_SFI_COMPVAL	The value

Table 21. Constants for XSF_SFI

Len	Type	Value	Name	Description
Data Types (XSF_SFI_DTYPE, XSF_SFI_DTYP1, XSF_SFI_DTYP2)				
1	HEX	00	XSF_SFI_DTYPE_UNDEF	Undefined data
1	HEX	01	XSF_SFI_DTYPE_CHAR_FIX	N byte character
1	HEX	02	XSF_SFI_DTYPE_BITFLAG	Bit flag byte (8 bits)
1	HEX	03	XSF_SFI_DTYPE_BIN8	1 byte (hex) value
1	HEX	04	XSF_SFI_DTYPE_BIN15	2 byte hex value
1	HEX	05	XSF_SFI_DTYPE_BIN31	4 byte hex value
1	HEX	06	XSF_SFI_DTYPE_BIN64	8 byte hex value
1	HEX	07	XSF_SFI_DTYPE_CHAR_VAR	Variable length character
1	HEX	08	XSF_SFI_DTYPE_COMPOUND	Compound SFI
1	HEX	09	XSF_SFI_DTYPE_JDATE	4 byte packed decimal date YYYYDDD

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	0A	XSF_SFI_DTYPE_TIME	4 byte packed decimal time HHMMSST
	Compound Types (XSF_SFI_CompType)			
1	HEX	00	XSF_SFI_COMPTYPE_UNDEF	Undefined type
1	HEX	01	XSF_SFI_COMPTYPE_FACTOR	Factored type
	Factors (XSF_SFI_Factor)			
1	HEX	00	XSF_SFI_FACTOR_BYTES	Value is in bytes
1	HEX	01	XSF_SFI_FACTOR_KB	Value is in kilobytes
1	HEX	02	XSF_SFI_FACTOR_MB	Value is in megabytes
1	HEX	03	XSF_SFI_FACTOR_GB	Value is in gigabytes
1	HEX	04	XSF_SFI_FACTOR_TB	Value is in terabytes
	Group SFIs Begin and End ACCESS			
8	HEX	0008021000000000	XSF_SFI_ACCESS	
3	HEX	021000	XSF_SFI_ID_ACCESS	
2	HEX	0008	XSF_ACCESS_LENGTH	
8	HEX	0008021080000000	XSF_SFI_EACCESS	
3	HEX	021080	XSF_SFI_ID_EACCESS	
2	HEX	0008	XSF_EACCESS_LENGTH	
	Begin and End ACTIONS			
8	HEX	0008022000000000	XSF_SFI_ACTIONS	
3	HEX	022000	XSF_SFI_ID_ACTIONS	
2	HEX	0008	XSF_ACTIONS_LENGTH	
8	HEX	0008022080000000	XSF_SFI_EACTIONS	
3	HEX	022080	XSF_SFI_ID_EACTIONS	
2	HEX	0008	XSF_EACTIONS_LENGTH	
	Begin and End CNTL			
8	HEX	0008024000000000	XSF_SFI_CNTL	
3	HEX	024000	XSF_SFI_ID_CNTL	
2	HEX	0008	XSF_CNTL_LENGTH	
8	HEX	0008024080000000	XSF_SFI_ECNTL	
3	HEX	024080	XSF_SFI_ID_ECNTL	
2	HEX	0008	XSF_ECNTL_LENGTH	
	Begin and End CONTROL			
8	HEX	0008025000000000	XSF_SFI_CONTROL	
3	HEX	025000	XSF_SFI_ID_CONTROL	
2	HEX	0008	XSF_CONTROL_LENGTH	
8	HEX	0008025080000000	XSF_SFI_ECONTROL	
3	HEX	025080	XSF_SFI_ID_ECONTROL	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0008	XSF_ECONTROL_LENGTH	
	Begin and End DATASET			
8	HEX	0008026000000000	XSF_SFI_DATASET	
3	HEX	026000	XSF_SFI_ID_DATASET	
2	HEX	0008	XSF_DATASET_LENGTH	
8	HEX	0008026080000000	XSF_SFI_EDATASET	
3	HEX	026080	XSF_SFI_ID_EDATASET	
2	HEX	0008	XSF_EDATASET_LENGTH	
	Begin and End DEFAULT			
8	HEX	0008026500000000	XSF_SFI_DEFAULT	
3	HEX	026500	XSF_SFI_ID_DEFAULT	
2	HEX	0008	XSF_DEFAULT_LENGTH	
8	HEX	0008026580000000	XSF_SFI_EDEFAULT	
3	HEX	026580	XSF_SFI_ID_EDEFAULT	
2	HEX	0008	XSF_EDEFAULT_LENGTH	
	Begin and End LOCDEF			
8	HEX	0008027000000000	XSF_SFI_LOCDEF	
3	HEX	027000	XSF_SFI_ID_LOCDEF	
2	HEX	0008	XSF_LOCDEF_LENGTH	
8	HEX	0008027080000000	XSF_SFI_ELOCDEF	
3	HEX	027080	XSF_SFI_ID_ELOCDEF	
2	HEX	0008	XSF_ELOCDEF_LENGTH	
	Begin and End MEDINF			
8	HEX	0008027500000000	XSF_SFI_MEDINF	
3	HEX	027500	XSF_SFI_ID_MEDINF	
2	HEX	0008	XSF_MEDINF_LENGTH	
8	HEX	0008027580000000	XSF_SFI_EMEDINF	
3	HEX	027580	XSF_SFI_ID_EMEDINF	
2	HEX	0008	XSF_EMEDINF_LENGTH	
	Begin and End MESSAGE			
8	HEX	0008028000000000	XSF_SFI_MESSAGE	
3	HEX	028000	XSF_SFI_ID_MESSAGE	
2	HEX	0008	XSF_MESSAGE_LENGTH	
8	HEX	0008028080000000	XSF_SFI_EMESSAGE	
3	HEX	028080	XSF_SFI_ID_EMESSAGE	
2	HEX	0008	XSF_EMESSAGE_LENGTH	
	Begin and End MNTMSG			

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	0008029000000000	XSF_SFI_MNTMSG	
3	HEX	029000	XSF_SFI_ID_MNTMSG	
2	HEX	0008	XSF_MNTMSG_LENGTH	
8	HEX	0008029080000000	XSF_SFI_EMNTMSG	
3	HEX	029080	XSF_SFI_ID_EMNTMSG	
2	HEX	0008	XSF_EMNTMSG_LENGTH	
		Begin and End MOVES		
8	HEX	000802A000000000	XSF_SFI_MOVES	
3	HEX	02A000	XSF_SFI_ID_MOVES	
2	HEX	0008	XSF_MOVES_LENGTH	
8	HEX	000802A080000000	XSF_SFI_EMOVES	
3	HEX	02A080	XSF_SFI_ID_EMOVES	
2	HEX	0008	XSF_EMOVES_LENGTH	
		Begin and End OPTION		
8	HEX	000802B000000000	XSF_SFI_OPTION	
3	HEX	02B000	XSF_SFI_ID_OPTION	
2	HEX	0008	XSF_OPTION_LENGTH	
8	HEX	000802B080000000	XSF_SFI_EOPTION	
3	HEX	02B080	XSF_SFI_ID_EOPTION	
2	HEX	0008	XSF_EOPTION_LENGTH	
		Begin and End OWNER		
8	HEX	000802C000000000	XSF_SFI_OWNER	
3	HEX	02C000	XSF_SFI_ID_OWNER	
2	HEX	0008	XSF_OWNER_LENGTH	
8	HEX	000802C080000000	XSF_SFI_EOWNER	
3	HEX	02C080	XSF_SFI_ID_EOWNER	
2	HEX	0008	XSF_EOWNER_LENGTH	
		Begin and End PRODUCT		
8	HEX	000802D000000000	XSF_SFI_PRODUCT	
3	HEX	02D000	XSF_SFI_ID_PRODUCT	
2	HEX	0008	XSF_PRODUCT_LENGTH	
8	HEX	000802D080000000	XSF_SFI_EPRODUCT	
3	HEX	02D080	XSF_SFI_ID_EPRODUCT	
2	HEX	0008	XSF_EPRODUCT_LENGTH	
		Begin and End RACK or BIN		
8	HEX	000802E000000000	XSF_SFI_RACK	
3	HEX	02E000	XSF_SFI_ID_RACK	
2	HEX	0008	XSF_RACK_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	000802E080000000	XSF_SFI_ERACK	
3	HEX	02E080	XSF_SFI_ID_ERACK	
2	HEX	0008	XSF_ERACK_LENGTH	
	Begin and End REJECT			
8	HEX	000802F000000000	XSF_SFI_REJECT	
3	HEX	02F000	XSF_SFI_ID_REJECT	
2	HEX	0008	XSF_REJECT_LENGTH	
8	HEX	000802F080000000	XSF_SFI_EREJECT	
3	HEX	02F080	XSF_SFI_ID_EREJECT	
2	HEX	0008	XSF_EREJECT_LENGTH	
	Begin and End SECCLS			
8	HEX	0008030000000000	XSF_SFI_SECCLS	
3	HEX	030000	XSF_SFI_ID_SECCLS	
2	HEX	0008	XSF_SECCLS_LENGTH	
8	HEX	0008030080000000	XSF_SFI_ESECCLS	
3	HEX	030080	XSF_SFI_ID_ESECCLS	
2	HEX	0008	XSF_ESECCLS_LENGTH	
	Begin and End SECLVL			
8	HEX	0008031000000000	XSF_SFI_SECLVL	
3	HEX	031000	XSF_SFI_ID_SECLVL	
2	HEX	0008	XSF_SECLVL_LENGTH	
8	HEX	0008031080000000	XSF_SFI_ESECLVL	
3	HEX	031080	XSF_SFI_ID_ESECLVL	
2	HEX	0008	XSF_ESECLVL_LENGTH	
	Begin and End STAT			
8	HEX	0008032000000000	XSF_SFI_STAT	
3	HEX	032000	XSF_SFI_ID_STAT	
2	HEX	0008	XSF_STAT_LENGTH	
8	HEX	0008032080000000	XSF_SFI_ESTAT	
3	HEX	032080	XSF_SFI_ID_ESTAT	
2	HEX	0008	XSF_ESTAT_LENGTH	
	Begin and End STORE			
8	HEX	0008033000000000	XSF_SFI_STORE	
3	HEX	033000	XSF_SFI_ID_STORE	
2	HEX	0008	XSF_STORE_LENGTH	
8	HEX	0008033080000000	XSF_SFI_ESTORE	
2	HEX	0008	XSF_ESTORE_LENGTH	
3	HEX	033080	XSF_SFI_ID_ESTORE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
	Begin and End SYSRETC			
8	HEX	0008034000000000	XSF_SFI_SYSRETC	
3	HEX	034000	XSF_SFI_ID_SYSRETC	
2	HEX	0008	XSF_SYSRETC_LENGTH	
8	HEX	0008034080000000	XSF_SFI_ESYSRETC	
3	HEX	034080	XSF_SFI_ID_ESYSRETC	
2	HEX	0008	XSF_ESYSRETC_LENGTH	
	Begin and End VLPOOL			
8	HEX	0008035000000000	XSF_SFI_VLPOOL	
3	HEX	035000	XSF_SFI_ID_VLPOOL	
2	HEX	0008	XSF_VLPOOL_LENGTH	
8	HEX	0008035080000000	XSF_SFI_EVLPOOL	
3	HEX	035080	XSF_SFI_ID_EVLPOOL	
2	HEX	0008	XSF_EVLPOOL_LENGTH	
	Begin and End VOL			
8	HEX	0008036000000000	XSF_SFI_VOLGRP	
3	HEX	036000	XSF_SFI_ID_VOL	
2	HEX	0008	XSF_VOLGRP_LENGTH	
8	HEX	0008036080000000	XSF_SFI_EVOLGRP	
3	HEX	036080	XSF_SFI_ID_EVOL	
2	HEX	0008	XSF_EVOLGRP_LENGTH	
	Begin and End VOLUME			
8	HEX	0008037000000000	XSF_SFI_VOLUME	
3	HEX	037000	XSF_SFI_ID_VOLUME	
2	HEX	0008	XSF_VOLUME_LENGTH	
8	HEX	0008037080000000	XSF_SFI_EVOLUME	
3	HEX	037080	XSF_SFI_ID_EVOLUME	
2	HEX	0008	XSF_EVOLUME_LENGTH	
	Begin and End VRS			
8	HEX	0008038000000000	XSF_SFI_VRSGRP	
3	HEX	038000	XSF_SFI_ID_VRS	
2	HEX	0008	XSF_VRSGRP_LENGTH	
8	HEX	0008038080000000	XSF_SFI_EVRSGRP	
3	HEX	038080	XSF_SFI_ID_EVRS	
2	HEX	0008	XSF_EVRSGRP_LENGTH	
	Begin and End PRODUCT VOLUME			
8	HEX	0008039000000000	XSF_SFI_PVOL	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	039000	XSF_SFI_ID_PVOL	
2	HEX	0008	XSF_PVOL_LENGTH	
8	HEX	0008039080000000	XSF_SFI_EPVOL	
3	HEX	039080	XSF_SFI_ID_EPVOL	
2	HEX	0008	XSF_EPVOL_LENGTH	
	Begin and End OPENRULE			
8	HEX	000803A000000000	XSF_SFI_OPENRULE	
3	HEX	03A000	XSF_SFI_ID_OPENRULE	
2	HEX	0008	XSF_OPENRULE_LENGTH	
8	HEX	000803A080000000	XSF_SFI_EOPENRULE	
3	HEX	03A080	XSF_SFI_ID_EOPENRULE	
2	HEX	0008	XSF_EOPENRULE_LENGTH	
	Begin and End PRTITION			
8	HEX	000803B000000000	XSF_SFI_PRTITION	
3	HEX	03B000	XSF_SFI_ID_PRTITION	
2	HEX	0008	XSF_PRTITION_LENGTH	
8	HEX	000803B080000000	XSF_SFI_EPRTITION	
3	HEX	03B080	XSF_SFI_ID_EPRTITION	
2	HEX	0008	XSF_EPRTITION_LENGTH	
	Begin and End STATUS			
8	HEX	000803C000000000	XSF_SFI_STATUS	
3	HEX	03C000	XSF_SFI_ID_STATUS	
2	HEX	0008	XSF_STATUS_LENGTH	
8	HEX	000803C080000000	XSF_SFI_ESTATUS	
3	HEX	03C080	XSF_SFI_ID_ESTATUS	
2	HEX	0008	XSF_ESTATUS_LENGTH	
	Begin and End TASKS			
8	HEX	000803D000000000	XSF_SFI_TASKS	
3	HEX	03D000	XSF_SFI_ID_TASKS	
2	HEX	0008	XSF_TASKS_LENGTH	
8	HEX	000803D080000000	XSF_SFI_ETASKS	
3	HEX	03D080	XSF_SFI_ID_ETASKS	
2	HEX	0008	XSF_ETASKS_LENGTH	
	Return and Reason Code SFI's			
8	HEX	000C400000000005	XSF_SFI_FRC	Function return code
3	HEX	400000	XSF_SFI_FRC_ID	
2	HEX	000C	XSF_FRC_LENGTH	
1	HEX	05	XSF_FRC_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	000C401000000005	XSF_SFI_FRS	Function reason code
3	HEX	401000	XSF_SFI_FRS_ID	
2	HEX	000C	XSF_FRS_LENGTH	
1	HEX	05	XSF_FRS_DTYPE	
8	HEX	000C402000000005	XSF_SFI_RSNC	Reason code
3	HEX	402000	XSF_SFI_RSNC_ID	
2	HEX	000C	XSF_RSNC_LENGTH	
1	HEX	05	XSF_RSNC_DTYPE	
8	HEX	000C403000000005	XSF_SFI_RTNC	Return code
3	HEX	403000	XSF_SFI_RTNC_ID	
2	HEX	000C	XSF_RTNC_LENGTH	
1	HEX	05	XSF_RTNC_DTYPE	
8	HEX	0010404000000007	XSF_SFI_SVCN	Service name
3	HEX	404000	XSF_SFI_SVCN_ID	
2	HEX	0010	XSF_SVCN_LENGTH	
1	HEX	07	XSF_SVCN_DTYPE	
Messages and Message Variable SFIs				
8	HEX	0103051000000007	XSF_SFI_MSGL	Message line
3	HEX	051000	XSF_SFI_MSGL_ID	
2	HEX	0103	XSF_MSGL_LENGTH	
1	HEX	07	XSF_MSGL_DTYPE	
8	HEX	0010052000000001	XSF_SFI_MSGN	Message number
3	HEX	052000	XSF_SFI_MSGN_ID	
2	HEX	0010	XSF_MSGN_LENGTH	
1	HEX	01	XSF_MSGN_DTYPE	
8	HEX	000C053000000005	XSF_SFI_ENTN	Number of entries
3	HEX	053000	XSF_SFI_ENTN_ID	
2	HEX	000C	XSF_ENTN_LENGTH	
1	HEX	05	XSF_ENTN_DTYPE	
8	HEX	0041054000000007	XSF_SFI_KEYF	From key
3	HEX	054000	XSF_SFI_KEYF_ID	
2	HEX	0041	XSF_KEYF_LENGTH	
1	HEX	07	XSF_KEYF_DTYPE	
8	HEX	0041054200000007	XSF_SFI_KEYT	To key
3	HEX	054200	XSF_SFI_KEYT_ID	
2	HEX	0041	XSF_KEYT_LENGTH	
1	HEX	07	XSF_KEYT_DTYPE	
8	HEX	0010055000000007	XSF_SFI_TYPF	From type
3	HEX	055000	XSF_SFI_TYPF_ID	
2	HEX	0010	XSF_TYPF_LENGTH	
1	HEX	07	XSF_TYPF_DTYPE	
8	HEX	0010055200000007	XSF_SFI_TYPT	To type

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	055200	XSF_SFI_TYPT_ID	
2	HEX	0010	XSF_TYPT_LENGTH	
1	HEX	07	XSF_TYPT_DTYPE	
8	HEX	005C057000000007	XSF_SFI_CONT	Continue information
3	HEX	057000	XSF_SFI_CONT_ID	
2	HEX	005C	XSF_CONT_LENGTH	
1	HEX	07	XSF_CONT_DTYPE	
Output Data SFIs				
8	HEX	0009800500000003	XSF_SFI_ABND	Closed by ABEND
3	HEX	800500	XSF_SFI_ABND_ID	
2	HEX	0009	XSF_ABND_LENGTH	
1	HEX	03	XSF_ABND_DTYPE	
1	NUMB HEX	00	XSF_ABND_DATA_NO	
1	NUMB HEX	01	XSF_ABND_DATA_YES	
8	HEX	0009800800000003	XSF_SFI_ACCT	Accounting source
3	HEX	800800	XSF_SFI_ACCT_ID	
2	HEX	0009	XSF_ACCT_LENGTH	
1	HEX	03	XSF_ACCT_DTYPE	
1	NUMB HEX	00	XSF_ACCT_DATA_JOB	
1	NUMB HEX	01	XSF_ACCT_DATA_STEP	
8	HEX	0030801000000007	XSF_SFI_ACN	Account number
3	HEX	801000	XSF_SFI_ACN_ID	
2	HEX	0030	XSF_ACN_LENGTH	
1	HEX	07	XSF_ACN_DTYPE	
8	HEX	0009801800000003	XSF_SFI_ACS	SMS ACS support
3	HEX	801800	XSF_SFI_ACS_ID	
2	HEX	0009	XSF_ACS_LENGTH	
1	HEX	03	XSF_ACS_DTYPE	
1	NUMB HEX	00	XSF_ACS_DATA_NO	
1	NUMB HEX	01	XSF_ACS_DATA_YES	
8	HEX	0009802000000002	XSF_SFI_ACT	Actions on release
3	HEX	802000	XSF_SFI_ACT_ID	
2	HEX	0009	XSF_ACT_LENGTH	
1	HEX	02	XSF_ACT_DTYPE	
1	HEX	80	XSF_ACT_FLAG_SCRATCH	
1	HEX	40	XSF_ACT_FLAG_REPLACE	
1	HEX	20	XSF_ACT_FLAG_INIT	
1	HEX	10	XSF_ACT_FLAG_ERASE	
1	HEX	08	XSF_ACT_FLAG_RETURN	
1	HEX	04	XSF_ACT_FLAG_NOTIFY	
8	HEX	0030803001000007	XSF_SFI_ADL	Address line
3	HEX	803001	XSF_SFI_ADL_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0030	XSF_ADL_LENGTH	
1	HEX	07	XSF_ADL_DTYPE	
8	HEX	000C804000000009	XSF_SFI_ADTJ	Assigned date
3	HEX	804000	XSF_SFI_ADTJ_ID	
2	HEX	000C	XSF_ADTJ_LENGTH	
1	HEX	09	XSF_ADTJ_DTYPE	
8	HEX	0009805000000002	XSF_SFI_AST	Action status
3	HEX	805000	XSF_SFI_AST_ID	
2	HEX	0009	XSF_AST_LENGTH	
1	HEX	02	XSF_AST_DTYPE	
1	HEX	80	XSF_AST_FLAG_PENDING	
1	HEX	40	XSF_AST_FLAG_CONFIRMED	
1	HEX	20	XSF_AST_FLAG_COMPLETE	
1	HEX	10	XSF_AST_FLAG_UNKNOWN	
8	HEX	000C806000000000A	XSF_SFI_ATM	Assigned time
3	HEX	806000	XSF_SFI_ATM_ID	
2	HEX	000C	XSF_ATM_LENGTH	
1	HEX	0A	XSF_ATM_DTYPE	
8	HEX	000A807000000004	XSF_SFI_AUD	SMF audit record number
3	HEX	807000	XSF_SFI_AUD_ID	
2	HEX	000A	XSF_AUD_LENGTH	
1	HEX	04	XSF_AUD_DTYPE	
8	HEX	0009808000000002	XSF_SFI_AVL	Volume availability
3	HEX	808000	XSF_SFI_AVL_ID	
2	HEX	0009	XSF_AVL_LENGTH	
1	HEX	02	XSF_AVL_DTYPE	
1	HEX	40	XSF_AVL_FLAG_PENDREL	
1	HEX	20	XSF_AVL_FLAG_VITALRCD	
1	HEX	08	XSF_AVL_FLAG_ONLOAN	
1	HEX	04	XSF_AVL_FLAG_OPEN	
8	HEX	000C809000000009	XSF_SFI_BDTJ	Last CDS backup date
3	HEX	809000	XSF_SFI_BDTJ_ID	
2	HEX	000C	XSF_BDTJ_LENGTH	
1	HEX	09	XSF_BDTJ_DTYPE	
8	HEX	000C809310000005	XSF_SFI_BESK	BES key index
3	HEX	809310	XSF_SFI_BESK_ID	
2	HEX	000C	XSF_BESK_LENGTH	
1	HEX	05	XSF_BESK_DTYPE	
8	HEX	000E80A000000001	XSF_SFI_BIN	Bin number
3	HEX	80A000	XSF_SFI_BIN_ID	
2	HEX	000E	XSF_BIN_LENGTH	
1	HEX	01	XSF_BIN_DTYPE	
8	HEX	001080B000000007	XSF_SFI_BKPP	Backup procedure name

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	80B000	XSF_SFI_BKPP_ID	
2	HEX	0010	XSF_BKPP_LENGTH	
1	HEX	07	XSF_BKPP_DTYPE	
8	HEX	000C80C000000005	XSF_SFI_BLKC	Block count
3	HEX	80C000	XSF_SFI_BLKC_ID	
2	HEX	000C	XSF_BLKC_LENGTH	
1	HEX	05	XSF_BLKC_DTYPE	
8	HEX	000C80D000000005	XSF_SFI_BLKS	Block size
3	HEX	80D000	XSF_SFI_BLKS_ID	
2	HEX	000C	XSF_BLKS_LENGTH	
1	HEX	05	XSF_BLKS_DTYPE	
8	HEX	000C80D030000005	XSF_SFI_BLKT	Total block count
3	HEX	80D030	XSF_SFI_BLKT_ID	
2	HEX	000C	XSF_BLKT_LENGTH	
1	HEX	05	XSF_BLKT_DTYPE	
8	HEX	001080D0B0000006	XSF_SFI_BLK6	
Total block count (8-byte (hex) value)				
3	HEX	80D0B0	XSF_SFI_BLK6_ID	
2	HEX	0010	XSF_BLK6_LENGTH	
1	HEX	06	XSF_BLK6_DTYPE	
8	HEX	000980E000000003	XSF_SFI_BLP	BLP option
3	HEX	80E000	XSF_SFI_BLP_ID	
2	HEX	0009	XSF_BLP_LENGTH	
1	HEX	03	XSF_BLP_DTYPE	
1	NUMB HEX	00	XSF_BLP_DATA_RMM	
1	NUMB HEX	01	XSF_BLP_DATA_NORMM	
8	HEX	001080F000000007	XSF_SFI_BMN	Bin number media name
3	HEX	80F000	XSF_SFI_BMN_ID	
2	HEX	0010	XSF_BMN_LENGTH	
1	HEX	07	XSF_BMN_DTYPE	
8	HEX	000C81000000000A	XSF_SFI_BTM	Last CDS backup time
3	HEX	810000	XSF_SFI_BTM_ID	
2	HEX	000C	XSF_BTM_LENGTH	
1	HEX	0A	XSF_BTM_DTYPE	
8	HEX	0009811000000002	XSF_SFI_CACT	Control active functions
3	HEX	811000	XSF_SFI_CACT_ID	
2	HEX	0009	XSF_CACT_LENGTH	
1	HEX	02	XSF_CACT_DTYPE	
1	HEX	80	XSF_CACT_FLAG_BACKUP	
1	HEX	40	XSF_CACT_FLAG_RESTORE	
1	HEX	20	XSF_CACT_FLAG_VERIFY	
1	HEX	10	XSF_CACT_FLAG_EXPROC	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	08	XSF_CACT_FLAG_EXTRACT	
1	HEX	04	XSF_CACT_FLAG_DSTORE	
1	HEX	02	XSF_CACT_FLAG_VRSEL	
8	HEX	0009811800000003	XSF_SFI_CATS	CATSYSID value
3	HEX	811800	XSF_SFI_CATS_ID	
2	HEX	0009	XSF_CATS_LENGTH	
1	HEX	03	XSF_CATA_DTYPE	
1	HEX	00	XSF_CATS_DATA_SET	
1	HEX	01	XSF_CATS_DATA_NOTSET	
1	HEX	02	XSF_CATS_DATA_STAR	
8	HEX	0010812000000007	XSF_SFI_CDS	Control data set ID
3	HEX	812000	XSF_SFI_CDS_ID	
2	HEX	0010	XSF_CDS_LENGTH	
1	HEX	07	XSF_CDS_DTYPE	
8	HEX	000A812400000004	XSF_SFI_CDSF	CDSFULL parmlib value
3	HEX	812400	XSF_SFI_CDSF_ID	
2	HEX	000A	XSF_CDSF_LENGTH	
1	HEX	04	XSF_CDSF_DTYPE	
8	HEX	0009812900000003	XSF_SFI_CDSQ	CDS ID enqueue name enabled
3	HEX	812900	XSF_SFI_CDSQ_ID	
2	HEX	0009	XSF_CDSQ_LENGTH	
1	HEX	03	XSF_CDSQ_DTYPE	
1	NUMB HEX	00	XSF_CDSQ_DATA_DISABLED	
1	NUMB HEX	01	XSF_CDSQ_DATA_ENABLED	
8	HEX	000A812A00000004	XSF_SFI_CDSU	CDS percentage used
3	HEX	812A00	XSF_SFI_CDSU_ID	
2	HEX	000A	XSF_CDSU_LENGTH	
1	HEX	04	XSF_CDSU_DTYPE	
8	HEX	000C813000000009	XSF_SFI_CDTJ	Create date
3	HEX	813000	XSF_SFI_CDTJ_ID	
2	HEX	000C	XSF_CDTJ_LENGTH	
1	HEX	09	XSF_CDTJ_DTYPE	
8	HEX	0010814000000007	XSF_SFI_CJBN	Job name
3	HEX	814000	XSF_SFI_CJBN_ID	
2	HEX	0010	XSF_CJBN_LENGTH	
1	HEX	07	XSF_CJBN_DTYPE	
8	HEX	0010815000000007	XSF_SFI_CLIB	Current library name
3	HEX	815000	XSF_SFI_CLIB_ID	
2	HEX	0010	XSF_CLIB_LENGTH	
1	HEX	07	XSF_CLIB_DTYPE	
8	HEX	0028816000000007	XSF_SFI_CLS	Security class description
3	HEX	816000	XSF_SFI_CLS_ID	
2	HEX	0028	XSF_CLS_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	07	XSF_CLS_DTYPE	
8	HEX	0009816900000003	XSF_SFI_CMDD	COMMANDAUTH data set name
3	HEX	816900	XSF_SFI_CMDD_ID	
2	HEX	0009	XSF_CMDD_LENGTH	
1	HEX	03	XSF_CMDD_DTYPE	
1	NUMB HEX	00	XSF_CMDD_DATA_NO	
1	NUMB HEX	01	XSF_CMDD_DATA_YES	
8	HEX	00098169A0000003	XSF_SFI_CMDO	COMMANDAUTH owner
3	HEX	8169A0	XSF_SFI_CMDO_ID	
2	HEX	0009	XSF_CMDO_LENGTH	
1	HEX	03	XSF_CMDO_DTYPE	
1	NUMB HEX	00	XSF_CMDO_DATA_NO	
1	NUMB HEX	01	XSF_CMDO_DATA_YES	
8	HEX	000C817000000005	XSF_SFI_CNT	Bin, rack or volume count
3	HEX	817000	XSF_SFI_CNT_ID	
2	HEX	000C	XSF_CNT_LENGTH	
1	HEX	05	XSF_CNT_DTYPE	
8	HEX	0010817820000001	XSF_SFI_CPGM	Creating program name
3	HEX	817820	XSF_SFI_CPGM_ID	
2	HEX	0010	XSF_CPGM_LENGTH	
1	HEX	01	XSF_CPGM_DTYPE	
8	HEX	000C817890000005	XSF_SFI_CRAT	
Compression ratio in hundreths				
3	HEX	817890	XSF_SFI_CRAT_ID	
2	HEX	000C	XSF_CRAT_LENGTH	
1	HEX	05	XSF_CRAT_DTYPE	
8	HEX	0010817900000007	XSF_SFI_CRID	Creating user ID
3	HEX	817900	XSF_SFI_CRID_ID	
2	HEX	0010	XSF_CRID_LENGTH	
1	HEX	07	XSF_CRID_DTYPE	
8	HEX	000C818000000005	XSF_SFI_CRP	CATRETPD retention period
3	HEX	818000	XSF_SFI_CRP_ID	
2	HEX	000C	XSF_CRP_LENGTH	
1	HEX	05	XSF_CRP_DTYPE	
8	HEX	000C818800000009	XSF_SFI_CSDT	Catalog synchronize date
3	HEX	818800	XSF_SFI_CSDT_ID	
2	HEX	000C	XSF_CSDT_LENGTH	
1	HEX	09	XSF_CSDT_DTYPE	
8	HEX	0047819200000007	XSF_SFI_CSHN	Client/Server host name
3	HEX	819200	XSF_SFI_CSHN_ID	
2	HEX	0047	XSF_CSHN_LENGTH	
1	HEX	07	XSF_CSHN_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	0035819250000007	XSF_SFI_CSIP	Client/Server IP address
3	HEX	819250	XSF_SFI_CSIP_ID	
2	HEX	0035	XSF_CSIP_LENGTH	
1	HEX	07	XSF_CSIP_DTYPE	
8	HEX	0010819000000007	XSF_SFI_CSG	Current storage group
3	HEX	819000	XSF_SFI_CSG_ID	
2	HEX	0010	XSF_CSG_LENGTH	
1	HEX	07	XSF_CSG_DTYPE	
8	HEX	000C81940000000A	XSF_SFI_CSTM	Catalog synchronize time
3	HEX	819400	XSF_SFI_CSTM_ID	
2	HEX	000C	XSF_CSTM_LENGTH	
1	HEX	0A	XSF_CSTM_DTYPE	
8	HEX	0009819600000003	XSF_SFI_CSVE	Stacked volume enabled status
3	HEX	819600	XSF_SFI_CSVE_ID	
2	HEX	0009	XSF_CSVE_LENGTH	
1	HEX	03	XSF_CSVE_DTYPE	
1	NUMB HEX	00	XSF_CSVE_DATA_NONE	
1	NUMB HEX	01	XSF_CSVE_DATA_ENABLED	
1	NUMB HEX	02	XSF_CSVE_DATA_DISABLED	
1	NUMB HEX	03	XSF_CSVE_DATA_MIXED	
8	HEX	000C819785000005	XSF_SFI_CTLD	Catalog Days
3	HEX	819785	XSF_SFI_CTLD_ID	
2	HEX	000C	XSF_CTLD_LENGTH	
1	HEX	05	XSF_CTLD_DTYPE	
8	HEX	0009819800000003	XSF_SFI_CTLG	Catalog status
3	HEX	819800	XSF_SFI_CTLG_ID	
2	HEX	0009	XSF_CTLG_LENGTH	
1	HEX	03	XSF_CTLG_DTYPE	
1	NUMB HEX	00	XSF_CTLG_DATA_UNKNOWN	
1	NUMB HEX	01	XSF_CTLG_DATA_NO	
1	NUMB HEX	02	XSF_CTLG_DATA_YES	
8	HEX	000C81A00000000A	XSF_SFI_CTM	Create time
3	HEX	81A000	XSF_SFI_CTM_ID	
2	HEX	000C	XSF_CTM_LENGTH	
1	HEX	0A	XSF_CTM_DTYPE	
8	HEX	001881A300000007	XSF_SFI_CTNR	Container
3	HEX	81A300	XSF_SFI_CTNR_ID	
2	HEX	0018	XSF_CTNR_LENGTH	
1	HEX	07	XSF_CTNR_DTYPE	
8	HEX	000981A400000003	XSF_SFI_CTRT	Catalog Retained
3	HEX	81A400	XSF_SFI_CTRT_ID	
2	HEX	0009	XSF_CTRT_LENGTH	
1	HEX	03	XSF_CTRT_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	00	XSF_CTRT_DATA_NO	
1	NUMB HEX	01	XSF_CTRT_DATA_KBC	
1	NUMB HEX	02	XSF_CTRT_DATA_UX	
1	NUMB HEX	03	XSF_CTRT_DATA_EXP	
8	HEX	000C81B000000005	XSF_SFI_DBN	Bin numbers in DISTANT
3	HEX	81B000	XSF_SFI_DBN_ID	
2	HEX	000C	XSF_DBN_LENGTH	
1	HEX	05	XSF_DBN_DTYPE	
8	HEX	000E81A600000001	XSF_SFI_DBIN	Destination bin number
3	HEX	81A600	XSF_SFI_DBIN_ID	
2	HEX	000E	XSF_DBIN_LENGTH	
1	HEX	01	XSF_DBIN_DTYPE	
8	HEX	001081A700000007	XSF_SFI_DBMN	Destination bin media name
3	HEX	81A700	XSF_SFI_DBMN_ID	
2	HEX	0010	XSF_DBMN_LENGTH	
1	HEX	07	XSF_DBMN_DTYPE	
8	HEX	001081C000000007	XSF_SFI_DC	Data class
3	HEX	81C000	XSF_SFI_DC_ID	
2	HEX	0010	XSF_DC_LENGTH	
1	HEX	07	XSF_DC_DTYPE	
8	HEX	001081D000000007	XSF_SFI_DD	DD name
3	HEX	81D000	XSF_SFI_DD_ID	
2	HEX	0010	XSF_DD_LENGTH	
1	HEX	07	XSF_DD_DTYPE	
8	HEX	000C81E000000009	XSF_SFI_DDTJ	Delete or store date
3	HEX	81E000	XSF_SFI_DDTJ_ID	
2	HEX	000C	XSF_DDTJ_LENGTH	
1	HEX	09	XSF_DDTJ_DTYPE	
8	HEX	000A81E050000004	XSF_SFI_DEFC	DEFTABLE/ Entries count
3	HEX	81E050	XSF_SFI_DEFC_ID	
2	HEX	000A	XSF_DEFC_LENGTH	
1	HEX	04	XSF_DEFC_DTYPE	
8	HEX	003481E100000007	XSF_SFI_DEFD	DEFTABLE/Data set name
3	HEX	81E100	XSF_SFI_DEFD_ID	
2	HEX	0034	XSF_DEFD_LENGTH	
1	HEX	07	XSF_DEFD_DTYPE	
8	HEX	000C81E200000005	XSF_SFI_DEFE	
DEFTABLE/EXTRA RETENTION PERIOD				
3	HEX	81E200	XSF_SFI_DEFE_ID	
2	HEX	000C	XSF_DEFE_LENGTH	
1	HEX	05	XSF_DEFE_DTYPE	
8	HEX	001081E300000007	XSF_SFI_DEFJ	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
	DEFTABLE/JOBNAME			
3	HEX	81E300	XSF_SFI_DEFJ_ID	
2	HEX	0010	XSF_DEFJ_LENGTH	
1	HEX	07	XSF_DEFJ_DTYPE	
8	HEX	001181E400000007	XSF_SFI_DEFK	
	DEFTABLE/KEYDATE			
3	HEX	81E400	XSF_SFI_DEFK_ID	
2	HEX	0011	XSF_DEFK_LENGTH	
1	HEX	07	XSF_DEFK_DTYPE	
8	HEX	000C81E500000005	XSF_SFI_DEFL	
	DEFTABLE/LASTREF VALUE			
3	HEX	81E500	XSF_SFI_DEFL_ID	
2	HEX	000C	XSF_DEFL_LENGTH	
1	HEX	05	XSF_DEFL_DTYPE	
8	HEX	000981E550000003	XSF_SFI_DEFM	
	DEFTABLE/EDM			
3	HEX	81E550	XSF_SFI_DEFM_ID	
2	HEX	0009	XSF_DEFM_LENGTH	
1	HEX	03	XSF_DEFM_DTYPE	
1	NUMB HEX	FF	XSF_DEFM_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFM_DATA_NO	
1	NUMB HEX	01	XSF_DEFM_DATA_YES	
8	HEX	001081E570000007	XSF_SFI_DEFN	
	DEFTABLE/PROGRAM NAME			
3	HEX	81E570	XSF_SFI_DEFN_ID	
2	HEX	0010	XSF_DEFN_LENGTH	
1	HEX	07	XSF_DEFN_DTYPE	
8	HEX	000981E600000003	XSF_SFI_DEFO	
	DEFTABLE/RETENTION OVERRIDE			
3	HEX	81E600	XSF_SFI_DEFO_ID	
2	HEX	0009	XSF_DEFO_LENGTH	
1	HEX	03	XSF_DEFO_DTYPE	
1	NUMB HEX	FF	XSF_DEFO_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFO_DATA_NO	
1	NUMB HEX	01	XSF_DEFO_DATA_YES	
8	HEX	000E81E700000007	XSF_SFI_DEFP	
	DEFTABLE/POOL ID			

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	81E700	XSF_SFI_DEFP_ID	
2	HEX	000E	XSF_DEFP_LENGTH	
1	HEX	07	XSF_DEFP_DTYPE	
8	HEX	000981E800000003	XSF_SFI_DEFR	
	DEFTABLE/RETENTION METHOD			
3	HEX	81E800	XSF_SFI_DEFR_ID	
2	HEX	0009	XSF_DEFR_LENGTH	
1	HEX	03	XSF_DEFR_DTYPE	
1	NUMB HEX	FF	XSF_DEFR_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFR_DATA_VRSEL	
1	NUMB HEX	01	XSF_DEFR_DATA_EXPDT	
8	HEX	001081E900000007	XSF_SFI_DEFV	
	DEFTABLE/VRS VALUE			
3	HEX	81E900	XSF_SFI_DEFV_ID	
2	HEX	0010	XSF_DEFV_LENGTH	
1	HEX	07	XSF_DEFV_DTYPE	
8	HEX	000981EA00000003	XSF_SFI_DEFW	
	DEFTABLE/WHILECATALOG			
3	HEX	81EA00	XSF_SFI_DEFW_ID	
2	HEX	0009	XSF_DEFW_LENGTH	
1	HEX	03	XSF_DEFW_DTYPE	
1	NUMB HEX	FF	XSF_DEFW_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFW_DATA_OFF	
1	NUMB HEX	01	XSF_DEFW_DATA_ON	
1	NUMB HEX	02	XSF_DEFW_DATA_UNTILEXPIRED	
8	HEX	000981EB00000003	XSF_SFI_DEFX	
	DEFTABLE/VRSEL EXCLUDED			
3	HEX	81EB00	XSF_SFI_DEFX_ID	
2	HEX	0009	XSF_DEFX_LENGTH	
1	HEX	03	XSF_DEFX_DTYPE	
1	NUMB HEX	FF	XSF_DEFX_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFX_DATA_NO	
1	NUMB HEX	01	XSF_DEFX_DATA_YES	
8	HEX	000981EC00000003	XSF_SFI_DEFY	
	DEFTABLE/RETAINBY VALUE			
3	HEX	81EC00	XSF_SFI_DEFY_ID	
2	HEX	0009	XSF_DEFY_LENGTH	
1	HEX	03	XSF_DEFY_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	FF	XSF_DEFY_DATA_BLANK	
1	NUMB HEX	00	XSF_DEFY_DATA_VOLUME	
1	NUMB HEX	01	XSF_DEFY_DATA_FIRST	
1	NUMB HEX	02	XSF_DEFY_DATA_SET	
8	HEX	000981ED00000003	XSF_SFI_DEFZ	
DEFTABLE/CONT				
3	HEX	81ED00	XSF_SFI_DEFZ_ID	
2	HEX	0009	XSF_DEFZ_LENGTH	
1	HEX	03	XSF_DEFZ_DTYPE	
1	NUMB HEX	00	XSF_DEFZ_DATA_NO	
1	NUMB HEX	01	XSF_DEFZ_DATA_YES	
8	HEX	000981F000000003	XSF_SFI_DEN	Media density
3	HEX	81F000	XSF_SFI_DEN_ID	
2	HEX	0009	XSF_DEN_LENGTH	
1	HEX	03	XSF_DEN_DTYPE	
1	NUMB HEX	00	XSF_DEN_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_DEN_DATA_1600	
1	NUMB HEX	02	XSF_DEN_DATA_6250	
1	NUMB HEX	03	XSF_DEN_DATA_3480	
1	NUMB HEX	04	XSF_DEN_DATA_COMPACT	
8	HEX	0026820000000007	XSF_SFI_DESC	Volume or VRS description
3	HEX	820000	XSF_SFI_DESC_ID	
2	HEX	0026	XSF_DESC_LENGTH	
1	HEX	07	XSF_DESC_DTYPE	
8	HEX	0010821000000007	XSF_SFI_DEST	Destination name
3	HEX	821000	XSF_SFI_DEST_ID	
2	HEX	0010	XSF_DEST_LENGTH	
1	HEX	07	XSF_DEST_DTYPE	
8	HEX	000C822000000001	XSF_SFI_DEV	Device number
3	HEX	822000	XSF_SFI_DEV_ID	
2	HEX	000C	XSF_DEV_LENGTH	
1	HEX	01	XSF_DEV_DTYPE	
8	HEX	000C822500000005	XSF_SFI_DKBC	Datasets Kept By Catalog
3	HEX	822500	XSF_SFI_DKBC_ID	
2	HEX	000C	XSF_DKBC_LENGTH	
1	HEX	05	XSF_DKBC_DTYPE	
8	HEX	000C823000000009	XSF_SFI_DLRJ	Date last read
3	HEX	823000	XSF_SFI_DLRJ_ID	
2	HEX	000C	XSF_DLRJ_LENGTH	
1	HEX	09	XSF_DLRJ_DTYPE	
8	HEX	0009823700000003	XSF_SFI_DLTD	Deleted by disposition
3	HEX	822700	XSF_SFI_DLTD_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0009	XSF_DLTD_LENGTH	
1	HEX	03	XSF_DLTD_DTYPE	
1	NUMB HEX	00	XSF_DLTD_DATA_NO	
1	NUMB HEX	01	XSF_DLTD_DATA_YES	
8	HEX	000C824000000009	XSF_SFI_DLWJ	Date last written
3	HEX	824000	XSF_SFI_DLWJ_ID	
2	HEX	000C	XSF_DLWJ_LENGTH	
1	HEX	09	XSF_DLWJ_DTYPE	
8	HEX	0034825000000007	XSF_SFI_DNM	Data set name mask
3	HEX	825000	XSF_SFI_DNM_ID	
2	HEX	0034	XSF_DNM_LENGTH	
1	HEX	07	XSF_DNM_DTYPE	
8	HEX	0009825E00000003	XSF_SFI_DPCT	Percent of volume
3	HEX	825E00	XSF_SFI_DPCT_ID	
2	HEX	0009	XSF_DPCT_LENGTH	
1	HEX	03	XSF_DPCT_DTYPE	
8	HEX	0030826000000007	XSF_SFI_DPT	Owner's department
3	HEX	826000	XSF_SFI_DPT_ID	
2	HEX	0030	XSF_DPT_LENGTH	
1	HEX	07	XSF_DPT_DTYPE	
8	HEX	000C827000000005	XSF_SFI_DRP	Default retention period
3	HEX	827000	XSF_SFI_DRP_ID	
2	HEX	000C	XSF_DRP_LENGTH	
1	HEX	05	XSF_DRP_DTYPE	
8	HEX	000C828000000005	XSF_SFI_DSC	Data set count
3	HEX	828000	XSF_SFI_DSC_ID	
2	HEX	000C	XSF_DSC_LENGTH	
1	HEX	05	XSF_DSC_DTYPE	
8	HEX	000C829000000005	XSF_SFI_DSEQ	Data set sequence
3	HEX	829000	XSF_SFI_DSEQ_ID	
2	HEX	000C	XSF_DSEQ_LENGTH	
1	HEX	05	XSF_DSEQ_DTYPE	
8	HEX	000A829500000001	XSF_SFI_DSFX	Active DEFAULTS table suffix
3	HEX	829500	XSF_SFI_DSFX_ID	
2	HEX	000A	XSF_DSFX_LENGTH	
1	HEX	01	XSF_DSFX_DTYPE	
8	HEX	003482A000000007	XSF_SFI_DSN	Data set name
3	HEX	82A000	XSF_SFI_DSN_ID	
2	HEX	0034	XSF_DSN_LENGTH	
1	HEX	07	XSF_DSN_DTYPE	
8	HEX	001082A500000007	XSF_SFI_DSPD	Disposition DD name
3	HEX	82A500	XSF_SFI_DSPD_ID	
2	HEX	0010	XSF_DSPD_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	07	XSF_DSPD_DTYPE	
8	HEX	001082AA00000007	XSF_SFI_DSPM	Disposition message prefix
3	HEX	82AA00	XSF_SFI_DSPM_ID	
2	HEX	0010	XSF_DSPM_LENGTH	
1	HEX	07	XSF_DSPM_DTYPE	
8	HEX	000982B000000003	XSF_SFI_DSR	Data set recording
3	HEX	82B000	XSF_SFI_DSR_ID	
2	HEX	0009	XSF_DSR_LENGTH	
1	HEX	03	XSF_DSR_DTYPE	
1	NUMB HEX	00	XSF_DSR_DATA_OFF	
1	NUMB HEX	01	XSF_DSR_DATA_ON	
8	HEX	001682B030000008	XSF_SFI_DSS6	Data set size
3	HEX	82B030	XSF_SFI_DSS6_ID	
2	HEX	0016	XSF_DSS6_LENGTH	
1	HEX	08	XSF_DSS6_DTYPE	Compound data
6	HEX	010303010A06	XSF_SFI_DSS6_COMP	
1	HEX	01	XSF_SFI_DSS6_COMPTYPE	
1	HEX	03	XSF_SFI_DSS6_COMPLEN1	
1	HEX	03	XSF_SFI_DSS6_COMPTYP1	
1	HEX	01	XSF_SFI_DSS6_COMPDAT1	
1	HEX	0A	XSF_SFI_DSS6_COMPLEN2	
1	HEX	06	XSF_SFI_DSS6_COMPTYP2	
8	HEX	000982B200000003	XSF_SFI_DSTT	Destination type
3	HEX	82B200	XSF_SFI_DSTT_ID	
2	HEX	0009	XSF_DSTT_LENGTH	
1	HEX	03	XSF_DSTT_DTYPE	
1	NUMB HEX	00	XSF_DSTT_DATA_SHELF	
1	NUMB HEX	01	XSF_DSTT_DATA_STORE_BUILTIN_BINS	
1	NUMB HEX	02	XSF_DSTT_DATA_MANUAL	
1	NUMB HEX	03	XSF_DSTT_DATA_AUTO	
1	NUMB HEX	04	XSF_DSTT_DATA_STORE_BINS	
1	NUMB HEX	05	XSF_DSTT_DATA_STORE_NOBINS	
8	HEX	000982B500000003	XSF_SFI_DSXP	DSNEXPIRE(NONE/BLOCK)
3	HEX	82B500	XSF_SFI_DSXP_ID	
2	HEX	0009	XSF_DSXP_LENGTH	
1	HEX	03	XSF_DSXP_DTYPE	
1	NUMB HEX	00	XSF_DSXP_DATA_NONE	
1	NUMB HEX	01	XSF_DSXP_DATA_BLOCK	
8	HEX	001082BB00000007	XSF_SFI_DSYS	Creation system ID for first file
3	HEX	82BB00	XSF_SFI_DSYS_ID	
2	HEX	0010	XSF_DSYS_LENGTH	
1	HEX	07	XSF_DSYS_DTYPE	
8	HEX	000982C000000003	XSF_SFI_DTE	Installation date format

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	82C000	XSF_SFI_DTE_ID	
2	HEX	0009	XSF_DTE_LENGTH	
1	HEX	03	XSF_DTE_DTYPE	
1	NUMB HEX	01	XSF_DTE_DATA_A	
1	NUMB HEX	02	XSF_DTE_DATA_E	
1	NUMB HEX	03	XSF_DTE_DATA_I	
1	NUMB HEX	04	XSF_DTE_DATA_J	
8	HEX	000C82D00000000A	XSF_SFI_DTM	Last store update run time
3	HEX	82D000	XSF_SFI_DTM_ID	
2	HEX	000C	XSF_DTM_LENGTH	
1	HEX	0A	XSF_DTM_DTYPE	
8	HEX	001082E000000007	XSF_SFI_EMN	Owner's node
3	HEX	82E000	XSF_SFI_EMN_ID	
2	HEX	0010	XSF_EMN_LENGTH	
1	HEX	07	XSF_EMN_DTYPE	
8	HEX	000982D500000003	XSF_SFI_EBIN	Extended bin enabled status
3	HEX	82D500	XSF_SFI_EBIN_ID	
2	HEX	0009	XSF_EBIN_LENGTH	
1	HEX	03	XSF_EBIN_DTYPE	
1	NUMB HEX	00	XSF_EBIN_DATA_DISABLED	
1	NUMB HEX	01	XSF_EBIN_DATA_ENABLED	
8	HEX	000982D700000003	XSF_SFI_EDM	Volume EDM attribute
3	HEX	82D700	XSF_SFI_EDM_ID	
2	HEX	0009	XSF_EDM_LENGTH	
1	HEX	03	XSF_EDM_DTYPE	
1	NUMB HEX	00	XSF_EDM_DATA_NO	
1	NUMB HEX	01	XSF_EDM_DATA_YES	
8	HEX	004782DFF0000007	XSF_SFI_EML	Owner's email address
3	HEX	82DFF0	XSF_SFI_EML_ID	
2	HEX	0047	XSF_EML_LENGTH	
1	HEX	07	XSF_EML_DTYPE	
8	HEX	001082F000000007	XSF_SFI_EMU	Owner's user ID
3	HEX	82F000	XSF_SFI_EMU_ID	
2	HEX	0010	XSF_EMU_LENGTH	
1	HEX	07	XSF_EMU_DTYPE	
8	HEX	001C830000000007	XSF_SFI_ETL	Owner's external phone number
3	HEX	830000	XSF_SFI_ETL_ID	
2	HEX	001C	XSF_ETL_LENGTH	
1	HEX	07	XSF_ETL_DTYPE	
8	HEX	0009830800000003	XSF_SFI_EXRB	EXPDT RetainBy
3	HEX	830800	XSF_SFI_EXRB_ID	
2	HEX	0009	XSF_EXRB_LENGTH	
1	HEX	03	XSF_EXRB_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	00	XSF_EXRB_VOLUME	
1	NUMB HEX	01	XSF_EXRB_FIRST	
1	NUMB HEX	02	XSF_EXRB_SET	
8	HEX	000C831000000007	XSF_SFI_FCD	Product feature code
3	HEX	831000	XSF_SFI_FCD_ID	
2	HEX	000C	XSF_FCD_LENGTH	
1	HEX	07	XSF_FCD_DTYPE	
8	HEX	0009831800000003	XSF_SFI_FCSP	Catalog synchronize in progress
3	HEX	831800	XSF_SFI_FCSP_ID	
2	HEX	0009	XSF_FCSP_LENGTH	
1	HEX	03	XSF_FCSP_DTYPE	
8	HEX	000C832000000005	XSF_SFI_FDB	Free bins in DISTANT location
3	HEX	832000	XSF_SFI_FDB_ID	
2	HEX	000C	XSF_FDB_LENGTH	
1	HEX	05	XSF_FDB_DTYPE	
8	HEX	000C833000000005	XSF_SFI_FILE	Physical file sequence
3	HEX	833000	XSF_SFI_FILE_ID	
2	HEX	000C	XSF_FILE_LENGTH	
1	HEX	05	XSF_FILE_DTYPE	
8	HEX	000C834000000005	XSF_SFI_FLB	Free bin numbers in LOCAL
3	HEX	834000	XSF_SFI_FLB_ID	
2	HEX	000C	XSF_FLB_LENGTH	
1	HEX	05	XSF_FLB_DTYPE	
8	HEX	001C835000000007	XSF_SFI_FOR	Owner's forename
3	HEX	835000	XSF_SFI_FOR_ID	
2	HEX	001C	XSF_FOR_LENGTH	
1	HEX	07	XSF_FOR_DTYPE	
8	HEX	000C836000000005	XSF_SFI_FRB	Free bin numbers in REMOTE
2	HEX	000C	XSF_FRB_LENGTH	
3	HEX	836000	XSF_SFI_FRB_ID	
1	HEX	05	XSF_FRB_DTYPE	
8	HEX	000C837000000005	XSF_SFI_FRK	Free rack numbers in library
3	HEX	837000	XSF_SFI_FRK_ID	
2	HEX	000C	XSF_FRK_LENGTH	
1	HEX	05	XSF_FRK_DTYPE	
8	HEX	0009837500000003	XSF_SFI_FRXP	FORCEEXPIRE
3	HEX	837500	XSF_SFI_FXRP_ID	
2	HEX	0009	XSF_FRXP_LENGTH	
1	HEX	03	XSF_FRXP_DTYPE	
1	NUMB HEX	00	XSF_FRXP_DATA_NO	
1	NUMB HEX	01	XSF_FRXP_DATA_YES	
8	HEX	0009837800000003	XSF_SFI_GDGC	GDG CycleBy
3	HEX	837800	XSF_SFI_GDGC_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0009	XSF_GDGC_LENGTH	
1	HEX	03	XSF_GDGC_DTYPE	
1	DECIMAL	0	XSF_GDGC_DATA_GENERATION	
1	DECIMAL	1	XSF_GDGC_DATA_CRDATE	
8	HEX	000C837900000005	XSF_SFI_GDRP	GDG Retention period
3	HEX	837900	XSF_SFI_GDRP_ID	
2	HEX	000C	XSF_GDRP_LENGTH	
1	HEX	05	XSF_GDRP_DTYPE	
8	HEX	0009837940000003	XSF_SFI_GWCT	GDG WHILECATALOG
3	HEX	837940	XSF_SFI_GWCT_ID	
2	HEX	0009	XSF_GWCT_LENGTH	
1	HEX	03	XSF_GWCT_DTYPE	
1	NUMB HEX	00	XSF_GWCT_DATA_OFF	
1	NUMB HEX	01	XSF_GWCT_DATA_ON	
1	NUMB HEX	02	XSF_GWCT_DATA_UNTILEXPIRED	
8	HEX	0009837805000003	XSF_SFI_GDGD	GDG duplicate
3	HEX	837800	XSF_SFI_GDGD_ID	
2	HEX	0009	XSF_GDGD_LENGTH	
1	HEX	03	XSF_GDGD_DTYPE	
1	DECIMAL	0	XSF_GDGD_DATA_BUMP	
1	DECIMAL	1	XSF_GDGD_DATA_DROP	
1	DECIMAL	2	XSF_GDGD_DATA_KEEP	
1	DECIMAL	3	XSF_GDGD_DATA_COUNT	
8	HEX	000E838000000001	XSF_SFI_GRK	Generic rack number
3	HEX	838000	XSF_SFI_GRK_ID	
2	HEX	000E	XSF_GRK_LENGTH	
1	HEX	01	XSF_GRK_DTYPE	
8	HEX	0009838F40000003	XSF_SFI_HLD	Volume HOLD attribute
3	HEX	838F40	XSF_SFI_HLD_ID	
2	HEX	0009	XSF_HLD_LENGTH	
1	HEX	03	XSF_HLD_DTYPE	
1	NUMB HEX	00	XSF_HLD_DATA_NO	
1	NUMB HEX	01	XSF_HLD_DATA_YES	
8	HEX	0010839000000007	XSF_SFI_HLOC	Home location
3	HEX	839000	XSF_SFI_HLOC_ID	
2	HEX	0010	XSF_HLOC_LENGTH	
1	HEX	07	XSF_HLOC_DTYPE	
8	HEX	0009839200000003	XSF_SFI_HLOT	Home location type
3	HEX	839200	XSF_SFI_HLOT_ID	
2	HEX	0009	XSF_HLOT_LENGTH	
1	HEX	03	XSF_HLOT_DTYPE	
1	NUMB HEX	00	XSF_HLOT_DATA_SHELF	
1	NUMB HEX	02	XSF_HLOT_DATA_MANUAL	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	03	XSF_HLOT_DATA_AUTO	
8	HEX	0009839600000000	XSF_SFI_HSKP	Housekeeping functionality
3	HEX	839600	XSF_SFI_HSKP_ID	
2	HEX	0009	XSF_HSKP_LENGTH	
1	HEX	03	XSF_HSKP_DTYPE	
1	NUMB HEX	00	XSF_HSKP_DATA_NO	Limited
1	NUMB HEX	01	XSF_HSKP_DATA_YES	All
8	HEX	000983A000000003	XSF_SFI_INTR	Volume intransit status
3	HEX	83A000	XSF_SFI_INTR_ID	
2	HEX	0009	XSF_INTR_LENGTH	
1	HEX	03	XSF_INTR_DTYPE	
1	NUMB HEX	00	XSF_INTR_DATA_NO	
1	NUMB HEX	01	XSF_INTR_DATA_YES	
8	HEX	000983B000000003	XSF_SFI_IPL	Data check required in IPL?
3	HEX	83B000	XSF_SFI_IPL_ID	
2	HEX	0009	XSF_IPL_LENGTH	
1	HEX	03	XSF_IPL_DTYPE	
1	NUMB HEX	00	XSF_IPL_DATA_NO	
1	NUMB HEX	01	XSF_IPL_DATA_YES	
8	HEX	000983B830000003	XSF_SFI_IRMM	IRMM use
3	HEX	83B830	XSF_SFI_IRMM_ID	
2	HEX	0009	XSF_IRMM_LENGTH	
1	HEX	03	XSF_IRMM_DTYPE	
1	NUMB HEX	00	XSF_IRMM_DATA_NO	
1	NUMB HEX	01	XSF_IRMM_DATA_YES	
8	HEX	001083C000000007	XSF_SFI_ITL	Owner's internal phone number
3	HEX	83C000	XSF_SFI_ITL_ID	
2	HEX	0010	XSF_ITL_LENGTH	
1	HEX	07	XSF_ITL_DTYPE	
8	HEX	000C83CA00000009	XSF_SFI_JBDT	Last journal backup date
3	HEX	83CA00	XSF_SFI_JBDT_ID	
2	HEX	000C	XSF_JBDT_LENGTH	
1	HEX	09	XSF_JBDT_DTYPE	
8	HEX	000C83CB0000000A	XSF_SFI_JBTM	Last journal backup time
3	HEX	83CB00	XSF_SFI_JBTM_ID	
2	HEX	000C	XSF_JBTM_LENGTH	
1	HEX	0A	XSF_JBTM_DTYPE	
8	HEX	003483D000000007	XSF_SFI_JDS	Journal name
3	HEX	83D000	XSF_SFI_JDS_ID	
2	HEX	0034	XSF_JDS_LENGTH	
1	HEX	07	XSF_JDS_DTYPE	
8	HEX	000A83E000000004	XSF_SFI_JRNF	JOURNALFULL parmlib value
3	HEX	83E000	XSF_SFI_JRNF_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	000A	XSF_JRNF_LENGTH	
1	HEX	04	XSF_JRNF_DTYPE	
8	HEX	000983EA00000003	XSF_SFI_JRNS	Journal status
3	HEX	83EA00	XSF_SFI_JRNS_ID	
2	HEX	0009	XSF_JRNS_LENGTH	
1	HEX	03	XSF_JRNS_DTYPE	
1	HEX	00	XSF_JRNS_DISABLED	
1	HEX	01	XSF_JRNS_ENABLED	
1	HEX	02	XSF_JRNS_LOCKED	
8	HEX	000983ED00000003	XSF_SFI_JRNT	Journal transaction
3	HEX	83ED00	XSF_SFI_JRNT_ID	
2	HEX	0009	XSF_JRNT_LENGTH	
1	HEX	03	XSF_JRNT_DTYPE	
1	NUMB HEX	00	XSF_JRNT_DATA_NO	
1	NUMB HEX	01	XSF_JRNT_DATA_YES	
8	HEX	000A83F000000004	XSF_SFI_JRNU	Journal percentage used
3	HEX	83F000	XSF_SFI_JRNU_ID	
2	HEX	000A	XSF_JRNU_LENGTH	
1	HEX	04	XSF_JRNU_DTYPE	
8	HEX	004883F500000007	XSF_SFI_KEL1	Encryption key label 1
3	HEX	83F500	XSF_SFI_KEL1_ID	
2	HEX	0048	XSF_KEL1_LENGTH	
1	HEX	07	XSF_KEL1_DTYPE	
8	HEX	004883F505000007	XSF_SFI_KEL2	Encryption key label 2
3	HEX	83F505	XSF_SFI_KEL2_ID	
2	HEX	0048	XSF_KEL2_LENGTH	
1	HEX	07	XSF_KEL2_DTYPE	
8	HEX	000D83F520000007	XSF_SFI_KEM1	Encoding mechanism 1
3	HEX	83F520	XSF_SFI_KEM1_ID	
2	HEX	000D	XSF_KEM1_LENGTH	
1	HEX	07	XSF_KEM1_DTYPE	
8	HEX	000D83F525000007	XSF_SFI_KEM2	Encoding mechanism 2
3	HEX	83F525	XSF_SFI_KEM2_ID	
2	HEX	000D	XSF_KEM2_LENGTH	
1	HEX	07	XSF_KEM2_DTYPE	
8	HEX	0009840000000002	XSF_SFI_LBL	Volume label
3	HEX	840000	XSF_SFI_LBL_ID	
2	HEX	0009	XSF_LBL_LENGTH	
1	HEX	02	XSF_LBL_DTYPE	
1	HEX	20	XSF_LBL_FLAG_NL	
1	HEX	10	XSF_LBL_FLAG_AL	
1	HEX	08	XSF_LBL_FLAG_SL	
1	HEX	02	XSF_LBL_FLAG_BLP	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	01	XSF_LBL_FLAG_UL	
8	HEX	000C841000000005	XSF_SFI_LBN	Bin numbers in LOCAL
3	HEX	841000	XSF_SFI_LBN_ID	
2	HEX	000C	XSF_LBN_LENGTH	
1	HEX	05	XSF_LBN_DTYPE	
8	HEX	000C841500000009	XSF_SFI_LCDJ	Last Change Date
3	HEX	841500	XSF_SFI_LCDJ_ID	
2	HEX	000C	XSF_LCDJ_LENGTH	
1	HEX	09	XSF_LCDJ_DTYPE	
8	HEX	0010842000000007	XSF_SFI_LCID	Last change user ID
3	HEX	842000	XSF_SFI_LCID_ID	
2	HEX	0010	XSF_LCID_LENGTH	
1	HEX	07	XSF_LCID_DTYPE	
8	HEX	0010842500000007	XSF_SFI_LCSI	Last change system ID
3	HEX	842500	XSF_SFI_LCSI_ID	
2	HEX	0010	XSF_LCSI_LENGTH	
1	HEX	07	XSF_LCSI_DTYPE	
8	HEX	000A843000000004	XSF_SFI_LCT	Default lines per page
3	HEX	843000	XSF_SFI_LCT_ID	
2	HEX	000A	XSF_LCT_LENGTH	
1	HEX	04	XSF_LCT_DTYPE	
8	HEX	000C843100000005	XSF_SFI_LCTK	Local tasks
3	HEX	843100	XSF_SFI_LCTK_ID	
2	HEX	000C	XSF_LCTK_LENGTH	
1	HEX	05	XSF_LCTK_DTYPE	
8	HEX	000C84350000000A	XSF_SFI_LCTM	Last change time
3	HEX	843500	XSF_SFI_LCTM_ID	
2	HEX	000C	XSF_LCTM_LENGTH	
1	HEX	0A	XSF_LCTM_DTYPE	
8	HEX	000C843600000009	XSF_SFI_LCUD	Last "user" change date
3	HEX	843600	XSF_SFI_LCUD_ID	
2	HEX	000C	XSF_LCUD_LENGTH	
1	HEX	09	XSF_LCUD_DTYPE	
8	HEX	000C84370000000A	XSF_SFI_LCUT	Last "user" change time
3	HEX	843700	XSF_SFI_LCUT_ID	
2	HEX	000C	XSF_LCUT_LENGTH	
1	HEX	0A	XSF_LCUT_DTYPE	
8	HEX	0010843B00000001	XSF_SFI_LDD	Last used DD name
3	HEX	843B00	XSF_SFI_LDD_ID	
2	HEX	0010	XSF_LDD_LENGTH	
1	HEX	01	XSF_LDD_DTYPE	
8	HEX	0009844000000003	XSF_SFI_LDDF	Location definition exists
3	HEX	844000	XSF_SFI_LDDF_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0009	XSF_LDDF_LENGTH	
1	HEX	03	XSF_LDDF_DTYPE	
1	NUMB HEX	00	XSF_LDDF_DATA_NO	
1	NUMB HEX	01	XSF_LDDF_DATA_YES	
8	HEX	000C845000000001	XSF_SFI_LDEV	Last drive
3	HEX	845000	XSF_SFI_LDEV_ID	
2	HEX	000C	XSF_LDEV_LENGTH	
1	HEX	01	XSF_LDEV_DTYPE	
8	HEX	0010846000000007	XSF_SFI_LDLC	Location name
3	HEX	846000	XSF_SFI_LDLC_ID	
2	HEX	0010	XSF_LDLC_LENGTH	
1	HEX	07	XSF_LDLC_DTYPE	
8	HEX	0009847000000003	XSF_SFI_LDLT	Location type
3	HEX	847000	XSF_SFI_LDLT_ID	
2	HEX	0009	XSF_LDLT_LENGTH	
1	HEX	03	XSF_LDLT_DTYPE	
1	NUMB HEX	00	XSF_LDLT_DATA_SHELF	
1	NUMB HEX	01	XSF_LDLT_DATA_AUTO	
1	NUMB HEX	02	XSF_LDLT_DATA_MANUAL	
1	NUMB HEX	03	XSF_LDLT_DATA_STORE	
1	NUMB HEX	04	XSF_LDLT_DATA_HSTORE	
8	HEX	0010848000000007	XSF_SFI_LDMN	Location media name
3	HEX	848000	XSF_SFI_LDMN_ID	
2	HEX	0010	XSF_LDMN_LENGTH	
1	HEX	07	XSF_LDMN_DTYPE	
8	HEX	0009849000000003	XSF_SFI_LDMT	Location management type
3	HEX	849000	XSF_SFI_LDMT_ID	
2	HEX	0009	XSF_LDMT_LENGTH	
1	HEX	03	XSF_LDMT_DTYPE	
1	NUMB HEX	00	XSF_LDMT_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_LDMT_DATA_BIN	
1	NUMB HEX	02	XSF_LDMT_DATA_NOBINS	
8	HEX	000C84A000000005	XSF_SFI_LDPR	Location priority
3	HEX	84A000	XSF_SFI_LDPR_ID	
2	HEX	000C	XSF_LDPR_LENGTH	
1	HEX	05	XSF_LDPR_DTYPE	
8	HEX	000984A100000003	XSF_SFI_LDAM	Location Automove
3	HEX	84A100	XSF_SFI_LDAM_ID	
2	HEX	0009	XSF_LDAM_LENGTH	
1	HEX	03	XSF_LDAM_DTYPE	
1	NUMB HEX	00	XSF_LDAM_DATA_NO	
1	NUMB HEX	01	XSF_LDAM_DATA_YES	
8	HEX	000884B000000007	XSF_SFI_LINE	Output data line

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	84B000	XSF_SFI_LINE_ID	
2	HEX	0008	XSF_LINE_LENGTH	
1	HEX	07	XSF_LINE_DTYPE	
8	HEX	001084B420000001	XSF_SFI_LJOB	Last used job name
3	HEX	84B420	XSF_SFI_LJOB_ID	
2	HEX	0010	XSF_LJOB_LENGTH	
1	HEX	01	XSF_LJOB_DTYPE	
8	HEX	001084C000000007	XSF_SFI_LOAN	Loan location
3	HEX	84C000	XSF_SFI_LOAN_ID	
2	HEX	0010	XSF_LOAN_LENGTH	
1	HEX	07	XSF_LOAN_DTYPE	
8	HEX	001084D000000007	XSF_SFI_LOC	Location
3	HEX	84D000	XSF_SFI_LOC_ID	
2	HEX	0010	XSF_LOC_LENGTH	
1	HEX	07	XSF_LOC_DTYPE	
8	HEX	000984E000000003	XSF_SFI_LOCT	Location type
3	HEX	84E000	XSF_SFI_LOCT_ID	
2	HEX	0009	XSF_LOCT_LENGTH	
1	HEX	03	XSF_LOCT_DTYPE	
1	NUMB HEX	00	XSF_LOCT_DATA_SHELF	
1	NUMB HEX	01	XSF_LOCT_DATA_STORE_BUILTIN_BINS	
1	NUMB HEX	02	XSF_LOCT_DATA_MANUAL	
1	NUMB HEX	03	XSF_LOCT_DATA_AUTO	
1	NUMB HEX	04	XSF_LOCT_DATA_STORE_BINS	
1	NUMB HEX	05	XSF_LOCT_DATA_STORE_NOBINS	
1	NUMB HEX	06	XSF_LOCT_DATA_INCTNR	
8	HEX	001084E760000001	XSF_SFI_LPGM	Last used program name
3	HEX	84E760	XSF_SFI_LPGM_ID	
2	HEX	0010	XSF_LPGM_LENGTH	
1	HEX	01	XSF_LPGM_DTYPE	
8	HEX	000C84F000000005	XSF_SFI_LRCL	Logical record length
3	HEX	84F000	XSF_SFI_LRCL_ID	
2	HEX	000C	XSF_LRCL_LENGTH	
1	HEX	05	XSF_LRCL_DTYPE	
8	HEX	000C84F800000005	XSF_SFI_LRED	LASTREF extra days
3	HEX	84F800	XSF_SFI_LRED_ID	
2	HEX	000C	XSF_LRED_LENGTH	
1	HEX	05	XSF_LRED_DTYPE	
8	HEX	000C850000000005	XSF_SFI_LRK	Number of library rack numbers
3	HEX	850000	XSF_SFI_LRK_ID	
2	HEX	000C	XSF_LRK_LENGTH	
1	HEX	05	XSF_LRK_DTYPE	
8	HEX	0010850370000001	XSF_SFI_LSTP	Last used step name

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	850370	XSF_SFI_LSTP_ID	
2	HEX	0010	XSF_LSTP_LENGTH	
1	HEX	01	XSF_LSTP_DTYPE	
8	HEX	0009850500000003	XSF_SFI_LVC	Current label version
3	HEX	850500	XSF_SFI_LVC_ID	
2	HEX	0009	XSF_LVC_LENGTH	
1	HEX	03	XSF_LVC_DTYPE	
1	NUMB HEX	00	XSF_LVC_DATA_NO	
1	NUMB HEX	01	XSF_LVC_DATA_VERSION1	
1	NUMB HEX	03	XSF_LVC_DATA_VERSION3	
1	NUMB HEX	04	XSF_LVC_DATA_VERSION4	
8	HEX	0009850A00000003	XSF_SFI_LVN	Required label version
3	HEX	850A00	XSF_SFI_LVN_ID	
2	HEX	0009	XSF_LVN_LENGTH	
1	HEX	03	XSF_LVN_DTYPE	
1	NUMB HEX	00	XSF_LVN_DATA_NO	
1	NUMB HEX	03	XSF_LVN_DATA_VERSION3	
1	NUMB HEX	04	XSF_LVN_DATA_VERSION4	
8	HEX	0010851000000007	XSF_SFI_MC	Management class
3	HEX	851000	XSF_SFI_MC_ID	
2	HEX	0010	XSF_MC_LENGTH	
1	HEX	07	XSF_MC_DTYPE	
8	HEX	0009851200000003	XSF_SFI_MCAT	Management class attributes
3	HEX	851200	XSF_SFI_MCAT_ID	
2	HEX	0009	XSF_MCAT_LENGTH	
1	HEX	03	XSF_MCAT_DTYPE	
1	NUMB HEX	00	XSF_MCAT_DATA_NONE	
1	NUMB HEX	01	XSF_MCAT_DATA_ALL	
1	NUMB HEX	02	XSF_MCAT_DATA_VRSELXDI	
8	HEX	0010851400000001	XSF_SFI_MDNF	Media information name
3	HEX	851400	XSF_SFI_MDNF_ID	
2	HEX	0010	XSF_MDNF_LENGTH	
1	HEX	01	XSF_MDNF_DTYPE	
8	HEX	000C851980000005	XSF_SFI_MDRA	Replace policy for age
3	HEX	851980	XSF_SFI_MDRA_ID	
2	HEX	000C	XSF_MDRA_LENGTH	
1	HEX	05	XSF_MDRA_DTYPE	
8	HEX	000C8519C0000005	XSF_SFI_MDRP	Replace policy for permanent errors
3	HEX	8519C0	XSF_SFI_MDRP_ID	
2	HEX	000C	XSF_MDRP_LENGTH	
1	HEX	05	XSF_MDRP_DTYPE	
8	HEX	000C8519E0000005	XSF_SFI_MDRT	Replace policy for temporary errors
3	HEX	8519E0	XSF_SFI_MDRT_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	000C	XSF_MDRT_LENGTH	
1	HEX	05	XSF_MDRT_DTYPE	
8	HEX	000C8519F0000005	XSF_SFI_MDRW	Replace policy for write mount count
3	HEX	8519F0	XSF_SFI_MDRW_ID	
2	HEX	000C	XSF_MDRW_LENGTH	
1	HEX	05	XSF_MDRW_DTYPE	
8	HEX	0010851A00000001	XSF_SFI_MDRX	External recording technology
3	HEX	851A00	XSF_SFI_MDRX_ID	
2	HEX	0010	XSF_MDRX_LENGTH	
1	HEX	01	XSF_MDRX_DTYPE	
8	HEX	0034852000000007	XSF_SFI_MDS	CDS data set name
3	HEX	852000	XSF_SFI_MDS_ID	
2	HEX	0034	XSF_MDS_LENGTH	
1	HEX	07	XSF_MDS_DTYPE	
8	HEX	000C853000000009	XSF_SFI_MDTJ	CDS create date
3	HEX	853000	XSF_SFI_MDTJ_ID	
2	HEX	000C	XSF_MDTJ_LENGTH	
1	HEX	09	XSF_MDTJ_DTYPE	
8	HEX	0010853400000001	XSF_SFI_MDTX	External media type
3	HEX	853400	XSF_SFI_MDTX_ID	
2	HEX	0010	XSF_MDTX_LENGTH	
1	HEX	01	XSF_MDTX_DTYPE	
8	HEX	0009854000000003	XSF_SFI_MEDA	Media special attributes
3	HEX	854000	XSF_SFI_MEDA_ID	
2	HEX	0009	XSF_MEDA_LENGTH	
1	HEX	03	XSF_MEDA_DTYPE	
1	NUMB HEX	00	XSF_MEDA_DATA_NONE	
1	NUMB HEX	01	XSF_MEDA_DATA_READCOMP	
8	HEX	0009855000000003	XSF_SFI_MEDC	Media compaction
3	HEX	855000	XSF_SFI_MEDC_ID	
2	HEX	0009	XSF_MEDC_LENGTH	
1	HEX	03	XSF_MEDC_DTYPE	
1	NUMB HEX	00	XSF_MEDC_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_MEDC_DATA_NO	
1	NUMB HEX	02	XSF_MEDC_DATA_YES	
8	HEX	0010856000000007	XSF_SFI_MEDN	Media name
3	HEX	856000	XSF_SFI_MEDN_ID	
2	HEX	0010	XSF_MEDN_LENGTH	
1	HEX	07	XSF_MEDN_DTYPE	
8	HEX	0009857000000003	XSF_SFI_MEDR	Media recording format
3	HEX	857000	XSF_SFI_MEDR_ID	
2	HEX	0009	XSF_MEDR_LENGTH	
1	HEX	03	XSF_MEDR_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	00	XSF_MEDR_DATA_NOTCART	
1	NUMB HEX	01	XSF_MEDR_DATA_18TRK	
1	NUMB HEX	02	XSF_MEDR_DATA_36TRK	
1	NUMB HEX	03	XSF_MEDR_DATA_128TRK	
1	NUMB HEX	04	XSF_MEDR_DATA_256TRK	
1	NUMB HEX	05	XSF_MEDR_DATA_384TRK	
1	NUMB HEX	06	XSF_MEDR_DATA_EFMT1	
1	NUMB HEX	07	XSF_MEDR_DATA_EFMT2	
1	NUMB HEX	08	XSF_MEDR_DATA_EEFMT2	
1	NUMB HEX	09	XSF_MEDR_DATA_EFMT3	
1	NUMB HEX	0A	XSF_MEDR_DATA_EEFMT3	
1	NUMB HEX	0B	XSF_MEDR_DATA_EFMT4	
1	NUMB HEX	0C	XSF_MEDR_DATA_EEFMT4	
8	HEX	0009858000000003	XSF_SFI_MEDT	Media type
3	HEX	858000	XSF_SFI_MEDT_ID	
2	HEX	0009	XSF_MEDT_LENGTH	
1	HEX	03	XSF_MEDT_DTYPE	
1	NUMB HEX	00	XSF_MEDT_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_MEDT_DATA_CST	
1	NUMB HEX	02	XSF_MEDT_DATA_ECCST	
1	NUMB HEX	03	XSF_MEDT_DATA_HPCT	
1	NUMB HEX	04	XSF_MEDT_DATA_EHPCT	
1	NUMB HEX	05	XSF_MEDT_DATA_MEDIA5	
1	NUMB HEX	05	XSF_MEDT_DATA_ETC	
1	NUMB HEX	06	XSF_MEDT_DATA_EWTC	
1	NUMB HEX	07	XSF_MEDT_DATA_EETC	
1	NUMB HEX	08	XSF_MEDT_DATA_EEWTC	
1	NUMB HEX	09	XSF_MEDT_DATA_EXTC	
1	NUMB HEX	0A	XSF_MEDT_DATA_EXWTC	
1	NUMB HEX	0B	XSF_MEDT_DATA_EATC	
1	NUMB HEX	0C	XSF_MEDT_DATA_EAWTC	
1	NUMB HEX	0D	XSF_MEDT_DATA_EAETC	
8	HEX	0010859000000007	XSF_SFI_MFR	Source location name
3	HEX	859000	XSF_SFI_MFR_ID	
2	HEX	0010	XSF_MFR_LENGTH	
1	HEX	07	XSF_MFR_DTYPE	
8	HEX	001485A000000007	XSF_SFI_MID	Mount message ID
3	HEX	85A000	XSF_SFI_MID_ID	
2	HEX	0014	XSF_MID_LENGTH	
1	HEX	07	XSF_MID_DTYPE	
8	HEX	000E85A500000001	XSF_SFI_MIV	Moving-In volume
3	HEX	85A500	XSF_SFI_MIV_ID	
2	HEX	000E	XSF_MIV_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	01	XSF_MIV_DTYPE	
8	HEX	000E85A900000001	XSF_SFI_MOV	Moving-Out volume
3	HEX	85A900	XSF_SFI_MOV_ID	
2	HEX	000E	XSF_MOV_LENGTH	
1	HEX	01	XSF_MOV_DTYPE	
8	HEX	000985B000000003	XSF_SFI_MOVM	Move mode
3	HEX	85B000	XSF_SFI_MOVM_ID	
2	HEX	0009	XSF_MOVM_LENGTH	
1	HEX	03	XSF_MOVM_DTYPE	
1	NUMB HEX	00	XSF_MOVM_DATA_AUTO	
1	NUMB HEX	01	XSF_MOVM_DATA_MANUAL	
8	HEX	000985C000000003	XSF_SFI_MOP	Master overwrite
3	HEX	85C000	XSF_SFI_MOP_ID	
2	HEX	0009	XSF_MOP_LENGTH	
1	HEX	03	XSF_MOP_DTYPE	
1	NUMB HEX	01	XSF_MOP_DATA_ADD	
1	NUMB HEX	02	XSF_MOP_DATA_LAST	
1	NUMB HEX	03	XSF_MOP_DATA_MATCH	
1	NUMB HEX	04	XSF_MOP_DATA_USER	
8	HEX	000C85D000000005	XSF_SFI_MRP	Maximum retention period
3	HEX	85D000	XSF_SFI_MRP_ID	
2	HEX	000C	XSF_MRP_LENGTH	
1	HEX	05	XSF_MRP_DTYPE	
8	HEX	000985E000000003	XSF_SFI_MSGF	Case of message text
3	HEX	85E000	XSF_SFI_MSGF_ID	
2	HEX	0009	XSF_MSGF_LENGTH	
1	HEX	03	XSF_MSGF_DTYPE	
1	NUMB HEX	00	XSF_MSGF_DATA_MIXED	
1	NUMB HEX	01	XSF_MSGF_DATA_UPPER	
8	HEX	000985F000000003	XSF_SFI_MST	Move status
3	HEX	85F000	XSF_SFI_MST_ID	
2	HEX	0009	XSF_MST_LENGTH	
1	HEX	03	XSF_MST_DTYPE	
1	NUMB HEX	00	XSF_MST_DATA_UNKNOWN	
1	NUMB HEX	01	XSF_MST_DATA_PENDING	
1	NUMB HEX	02	XSF_MST_DATA_CONFIRMED	
1	NUMB HEX	03	XSF_MST_DATA_COMPLETE	
8	HEX	000C860000000000A	XSF_SFI_MTM	CDS create time
3	HEX	860000	XSF_SFI_MTM_ID	
2	HEX	000C	XSF_MTM_LENGTH	
1	HEX	0A	XSF_MTM_DTYPE	
8	HEX	0010861000000007	XSF_SFI_MTO	Target location name
3	HEX	861000	XSF_SFI_MTO_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0010	XSF_MTO_LENGTH	
1	HEX	07	XSF_MTO_DTYPE	
8	HEX	0009862000000003	XSF_SFI_MTP	CDS type
3	HEX	862000	XSF_SFI_MTP_ID	
2	HEX	0009	XSF_MTP_LENGTH	
1	HEX	03	XSF_MTP_DTYPE	
1	NUMB HEX	00	XSF_MTP_DATA_MASTER	
8	HEX	0009862800000003	XSF_SFI_MTY	Move type
3	HEX	862800	XSF_SFI_MTY_ID	
2	HEX	0009	XSF_MTY_LENGTH	
1	HEX	03	XSF_MTY_DTYPE	
1	NUMB HEX	00	XSF_MTY_DATA_NOTRTS	
1	NUMB HEX	01	XSF_MTY_DATA_RTS	
8	HEX	0009862B00000003	XSF_SFI_MVBY	Move by
3	HEX	862B00	XSF_SFI_MVBY_ID	
2	HEX	0009	XSF_MVBY_LENGTH	
1	HEX	03	XSF_MVBY_DTYPE	
1	NUMB HEX	00	XSF_MVBY_DATA_VOLUME	
1	NUMB HEX	01	XSF_MVBY_DATA_SET	
8	HEX	0009863000000003	XSF_SFI_MVS	MVS use
3	HEX	863000	XSF_SFI_MVS_ID	
2	HEX	0009	XSF_MVS_LENGTH	
1	HEX	03	XSF_MVS_DTYPE	
1	NUMB HEX	00	XSF_MVS_DATA_NO	
1	NUMB HEX	01	XSF_MVS_DATA_YES	
8	HEX	000C864500000005	XSF_SFI_NDRP	Not-GDG Retention Period
3	HEX	864500	XSF_SFI_NDRP_ID	
2	HEX	000C	XSF_NDRP_LENGTH	
1	HEX	05	XSF_NDRP_DTYPE	
8	HEX	0010865000000007	XSF_SFI_NLOC	Required location
3	HEX	865000	XSF_SFI_NLOC_ID	
2	HEX	0010	XSF_NLOC_LENGTH	
1	HEX	07	XSF_NLOC_DTYPE	
8	HEX	0009865200000003	XSF_SFI_NLOT	Required location type
3	HEX	865200	XSF_SFI_NLOT_ID	
2	HEX	0009	XSF_NLOT_LENGTH	
1	HEX	03	XSF_NLOT_DTYPE	
1	NUMB HEX	00	XSF_NLOT_DATA_SHELF	
1	NUMB HEX	01	XSF_NLOT_DATA_STORE_BUILTIN_BINS	
1	NUMB HEX	02	XSF_NLOT_DATA_MANUAL	
1	NUMB HEX	03	XSF_NLOT_DATA_AUTO	
1	NUMB HEX	04	XSF_NLOT_DATA_STORE_BINS	
1	NUMB HEX	05	XSF_NLOT_DATA_STORE_NOBINS	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	0010866000000007	XSF_SFI_NME	Security class name
3	HEX	866000	XSF_SFI_NME_ID	
2	HEX	0010	XSF_NME_LENGTH	
1	HEX	07	XSF_NME_DTYPE	
8	HEX	0009866800000003	XSF_SFI_NOT	Notify
3	HEX	866800	XSF_SFI_NOT_ID	
2	HEX	0009	XSF_NOT_LENGTH	
1	HEX	03	XSF_NOT_DTYPE	
1	NUMB HEX	00	XSF_NOT_DATA_NO	
1	NUMB HEX	01	XSF_NOT_DATA_YES	
8	HEX	000E867000000001	XSF_SFI_NVL	Next volume
3	HEX	867000	XSF_SFI_NVL_ID	
2	HEX	000E	XSF_NVL_LENGTH	
1	HEX	01	XSF_NVL_DTYPE	
8	HEX	0010868000000007	XSF_SFI_NVRS	Next VRS name
3	HEX	868000	XSF_SFI_NVRS_ID	
2	HEX	0010	XSF_NVRS_LENGTH	
1	HEX	07	XSF_NVRS_DTYPE	
8	HEX	0009868500000003	XSF_SFI_NWCT	Not-GDG WHILECATALOG
3	HEX	868500	XSF_SFI_NWCT_ID	
2	HEX	0009	XSF_NWCT_LENGTH	
1	HEX	03	XSF_NWCT_DTYPE	
1	NUMB HEX	00	XSF_NWCT_DATA_OFF	
1	NUMB HEX	01	XSF_NWCT_DATA_ON	
1	NUMB HEX	02	XSF_NWCT_DATA_UNTILEXPIRED	
8	HEX	0009869000000003	XSF_SFI_OAC	Owner access
3	HEX	869000	XSF_SFI_OAC_ID	
2	HEX	0009	XSF_OAC_LENGTH	
1	HEX	03	XSF_OAC_DTYPE	
1	NUMB HEX	00	XSF_OAC_DATA_READ	
1	NUMB HEX	01	XSF_OAC_DATA_UPDATE	
1	NUMB HEX	02	XSF_OAC_DATA_ALTER	
8	HEX	001086A000000007	XSF_SFI_OBMN	Old bin number media name
3	HEX	86A000	XSF_SFI_OBMN_ID	
2	HEX	0010	XSF_OBMN_LENGTH	
1	HEX	07	XSF_OBMN_DTYPE	
8	HEX	000E86B000000001	XSF_SFI_OBN	Old bin number
3	HEX	86B000	XSF_SFI_OBN_ID	
2	HEX	000E	XSF_OBN_LENGTH	
1	HEX	01	XSF_OBN_DTYPE	
8	HEX	000986B800000003	XSF_SFI_OCE	Volume information recorded at O/C/EOV
3	HEX	86B800	XSF_SFI_OCE_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0009	XSF_OCE_LENGTH	
1	HEX	03	XSF_OCE_DTYPE	
1	NUMB HEX	00	XSF_OCE_DATA_NO	
1	NUMB HEX	01	XSF_OCE_DATA_YES	
8	HEX	001086C000000007	XSF_SFI_OLOC	Old location
3	HEX	86C000	XSF_SFI_OLOC_ID	
2	HEX	0010	XSF_OLOC_LENGTH	
1	HEX	07	XSF_OLOC_DTYPE	
8	HEX	001086C100000007	XSF_SFI_OLON	Old loan location
3	HEX	86C100	XSF_SFI_OLON_ID	
2	HEX	0010	XSF_OLON_LENGTH	
1	HEX	07	XSF_OLON_DTYPE	
8	HEX	000986C200000003	XSF_SFI_OLOT	Old location type
3	HEX	86C200	XSF_SFI_OLOT_ID	
2	HEX	0009	XSF_OLOT_LENGTH	
1	HEX	03	XSF_OLOT_DTYPE	
1	NUMB HEX	00	XSF_OLOT_DATA_SHELF	
1	NUMB HEX	01	XSF_OLOT_DATA_STORE_BUILTIN_BINS	
1	NUMB HEX	02	XSF_OLOT_DATA_MANUAL	
1	NUMB HEX	03	XSF_OLOT_DATA_AUTO	
1	NUMB HEX	04	XSF_OLOT_DATA_STORE_BINS	
1	NUMB HEX	05	XSF_OLOT_DATA_STORE_NOBINS	
1	NUMB HEX	06	XSF_OLOT_DATA_STORE_INCTNR	
8	HEX	000A86D000000004	XSF_SFI_OPL	Position of rack number or pool ID
3	HEX	86D000	XSF_SFI_OPL_ID	
2	HEX	000A	XSF_OPL_LENGTH	
1	HEX	04	XSF_OPL_DTYPE	
8	HEX	000986E000000003	XSF_SFI_OPM	Operating mode
3	HEX	86E000	XSF_SFI_OPM_ID	
2	HEX	0009	XSF_OPM_LENGTH	
1	HEX	03	XSF_OPM_DTYPE	
1	NUMB HEX	01	XSF_OPM_DATA_M	
1	NUMB HEX	02	XSF_OPM_DATA_R	
1	NUMB HEX	03	XSF_OPM_DATA_W	
1	NUMB HEX	04	XSF_OPM_DATA_P	
8	HEX	000986E8A0000003	XSF_SFI_ORIA	OPENRULE input action
3	HEX	86E8A0	XSF_SFI_ORIA_ID	
2	HEX	0009	XSF_ORIA_LENGTH	
1	HEX	03	XSF_ORIA_DTYPE	
1	NUMB HEX	00	XSF_ORIA_DATA_ACCEPT	
1	NUMB HEX	01	XSF_ORIA_DATA_IGNORE	
1	NUMB HEX	02	XSF_ORIA_DATA_REJECT	
8	HEX	000986E8A8000002	XSF_SFI_ORII	OPENRULE input IGNORE condition

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	86E8A8	XSF_SFI_ORII_ID	
2	HEX	0009	XSF_ORII_LENGTH	
1	HEX	02	XSF_ORII_DTYPE	
1	HEX	80	XSF_ORII_DATA_SPECIFIC	
1	HEX	40	XSF_ORII_DATA_NONSPECIFIC	
1	HEX	C0	XSF_ORII_DATA_ANY	
8	HEX	000986E8B8000002	XSF_SFI_ORIR	OPENRULE input reject condition
3	HEX	86E8B8	XSF_SFI_ORIR_ID	
2	HEX	0009	XSF_ORIR_LENGTH	
1	HEX	02	XSF_ORIR_DTYPE	
1	HEX	80	XSF_ORIR_DATA_SYSID	
1	HEX	40	XSF_ORIR_DATA_CATLG	
8	HEX	000986EA00000003	XSF_SFI_OROA	OPENRULE output action
3	HEX	86EA00	XSF_SFI_OROA_ID	
2	HEX	0009	XSF_OROA_LENGTH	
1	HEX	03	XSF_OROA_DTYPE	
1	NUMB HEX	00	XSF_OROA_DATA_ACCEPT	
1	NUMB HEX	01	XSF_OROA_DATA_IGNORE	
1	NUMB HEX	02	XSF_OROA_DATA_REJECT	
8	HEX	000986EA08000002	XSF_SFI_OROI	OPENRULE output ignore condition
3	HEX	86EA08	XSF_SFI_OROI_ID	
2	HEX	0009	XSF_OROI_LENGTH	
1	HEX	02	XSF_OROI_DTYPE	
1	HEX	80	XSF_OROI_DATA_SPECIFIC	
1	HEX	40	XSF_OROI_DATA_NONSPECIFIC	
1	HEX	C0	XSF_OROI_DATA_ANY	
8	HEX	000986EA18000002	XSF_SFI_OROR	OPENRULE output reject condition
3	HEX	86EA18	XSF_SFI_OROR_ID	
2	HEX	0009	XSF_OROR_LENGTH	
1	HEX	02	XSF_OROR_DTYPE	
1	HEX	80	XSF_OROR_DATA_SYSID	
1	HEX	40	XSF_OROR_DATA_CATLG	
8	HEX	000986EF08000003	XSF_SFI_ORTP	OPENRULE entry type
3	HEX	86EF08	XSF_SFI_ORTP_ID	
2	HEX	0009	XSF_ORTP_LENGTH	
1	HEX	03	XSF_ORTP_DTYPE	
1	NUMB HEX	00	XSF_ORTP_DATA_RMM	
1	NUMB HEX	01	XSF_ORTP_DATA_NORMM	
8	HEX	000E86EF80000007	XSF_SFI_ORVS	OPENRULE volume range start
3	HEX	86EF80	XSF_SFI_ORVS_ID	
2	HEX	000E	XSF_ORVS_LENGTH	
1	HEX	07	XSF_ORVS_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	000E86EF85000007	XSF_SFI_ORVL	OPENRULE volume serial number, specific or generic
3	HEX	86EF85	XSF_SFI_ORVL_ID	
2	HEX	000E	XSF_ORVL_LENGTH	
1	HEX	07	XSF_ORVL_DTYPE	
8	HEX	000E86EF8F000007	XSF_SFI_ORVE	OPENRULE volume range end
3	HEX	86EF8F	XSF_SFI_ORVE_ID	
2	HEX	000E	XSF_ORVE_LENGTH	
1	HEX	07	XSF_ORVE_DTYPE	
8	HEX	000A86F000000004	XSF_SFI_OVL	Position of volume serial
3	HEX	86F000	XSF_SFI_OVL_ID	
2	HEX	000A	XSF_OVL_LENGTH	
1	HEX	04	XSF_OVL_DTYPE	
8	HEX	000E86F500000001	XSF_SFI_OVOL	Old volume
3	HEX	86F500	XSF_SFI_OVOL_ID	
2	HEX	000E	XSF_OVOL_LENGTH	
1	HEX	01	XSF_OVOL_DTYPE	
8	HEX	0010870000000007	XSF_SFI_OWN	Owner
3	HEX	870000	XSF_SFI_OWN_ID	
2	HEX	0010	XSF_OWN_LENGTH	
1	HEX	07	XSF_OWN_DTYPE	
8	HEX	000C871000000009	XSF_SFI_OXDJ	Original expiration date
3	HEX	871000	XSF_SFI_OXDJ_ID	
2	HEX	000C	XSF_OXDJ_LENGTH	
1	HEX	09	XSF_OXDJ_DTYPE	
8	HEX	0009871800000003	XSF_SFI_PACS	Pre-ACS support
3	HEX	871800	XSF_SFI_PACS_ID	
2	HEX	0009	XSF_PACS_LENGTH	
1	HEX	03	XSF_PACS_DTYPE	
1	NUMB HEX	00	XSF_PACS_DATA_NO	
1	NUMB HEX	01	XSF_PACS_DATA_YES	
8	HEX	0009871E00000003	XSF_SFI_PDA	PDA state
3	HEX	871E00	XSF_SFI_PDA_ID	
2	HEX	0009	XSF_PDA_LENGTH	
1	HEX	03	XSF_PDA_DTYPE	
1	NUMB HEX	00	XSF_PDA_DATA_OFF	
1	NUMB HEX	01	XSF_PDA_DATA_ON	
1	NUMB HEX	02	XSF_PDA_DATA_NONE	
8	HEX	0009871E10000003	XSF_SFI_PDAC	PDA block count
3	HEX	871E10	XSF_SFI_PDAC_ID	
2	HEX	0009	XSF_PDAC_LENGTH	
1	HEX	03	XSF_PDAC_DTYPE	
8	HEX	0009871E30000003	XSF_SFI_PDAL	PDA log state

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	871E30	XSF_SFI_PDAL_ID	
2	HEX	0009	XSF_PDAL_LENGTH	
1	HEX	03	XSF_PDAL_DTYPE	
1	NUMB HEX	00	XSF_PDAL_DATA_OFF	
1	NUMB HEX	01	XSF_PDAL_DATA_ON	
8	HEX	0009871E90000003	XSF_SFI_PDAS	PDA block size
3	HEX	871E90	XSF_SFI_PDAS_ID	
2	HEX	0009	XSF_PDAS_LENGTH	
1	HEX	03	XSF_PDAS_DTYPE	
8	HEX	0030872000000007	XSF_SFI_PDS	Pool description
3	HEX	872000	XSF_SFI_PDS_ID	
2	HEX	0030	XSF_PDS_LENGTH	
1	HEX	07	XSF_PDS_DTYPE	
8	HEX	0028873000000007	XSF_SFI_PDSC	Product description
3	HEX	873000	XSF_SFI_PDSC_ID	
2	HEX	0028	XSF_PDSC_LENGTH	
1	HEX	07	XSF_PDSC_DTYPE	
8	HEX	0009874000000002	XSF_SFI_PEND	Actions pending
3	HEX	874000	XSF_SFI_PEND_ID	
2	HEX	0009	XSF_PEND_LENGTH	
1	HEX	02	XSF_PEND_DTYPE	
1	HEX	80	XSF_PEND_FLAG_SCRATCH	
1	HEX	40	XSF_PEND_FLAG_REPLACE	
1	HEX	20	XSF_PEND_FLAG_INIT	
1	HEX	10	XSF_PEND_FLAG_ERASE	
1	HEX	08	XSF_PEND_FLAG_RETURN	
1	HEX	04	XSF_PEND_FLAG_NOTIFY	
8	HEX	000E875000000007	XSF_SFI_PID	Pool prefix
3	HEX	875000	XSF_SFI_PID_ID	
2	HEX	000E	XSF_PID_LENGTH	
1	HEX	07	XSF_PID_DTYPE	
8	HEX	0010876000000007	XSF_SFI_PLN	Pool name
3	HEX	876000	XSF_SFI_PLN_ID	
2	HEX	0010	XSF_PLN_LENGTH	
1	HEX	07	XSF_PLN_DTYPE	
8	HEX	0026877000000007	XSF_SFI_PNME	Software product name
3	HEX	877000	XSF_SFI_PNME_ID	
2	HEX	0026	XSF_PNME_LENGTH	
1	HEX	07	XSF_PNME_DTYPE	
8	HEX	0010878000000007	XSF_SFI_PNUM	Software product number
3	HEX	878000	XSF_SFI_PNUM_ID	
2	HEX	0010	XSF_PNUM_LENGTH	
1	HEX	07	XSF_PNUM_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	000C879000000005	XSF_SFI_PRD	Permanent read errors
3	HEX	879000	XSF_SFI_PRD_ID	
2	HEX	000C	XSF_PRD_LENGTH	
1	HEX	05	XSF_PRD_DTYPE	
8	HEX	000987A000000003	XSF_SFI_PRF	Pool definition RACF option
3	HEX	87A000	XSF_SFI_PRF_ID	
2	HEX	0009	XSF_PRF_LENGTH	
1	HEX	03	XSF_PRF_DTYPE	
1	NUMB HEX	00	XSF_PRF_DATA_NO	
1	NUMB HEX	01	XSF_PRF_DATA_YES	
8	HEX	000C87B000000005	XSF_SFI_PRTY	Priority
3	HEX	87B000	XSF_SFI_PRTY_ID	
2	HEX	000C	XSF_PRTY_LENGTH	
1	HEX	05	XSF_PRTY_DTYPE	
8	HEX	000A87C000000001	XSF_SFI_PSFX	Parmlib member suffix
3	HEX	87C000	XSF_SFI_PSFX_ID	
2	HEX	000A	XSF_PSFX_LENGTH	
1	HEX	01	XSF_PSFX_DTYPE	
8	HEX	000A87C010000001	XSF_SFI_PSF2	Parmlib member suffix 2
3	HEX	87C010	XSF_SFI_PSF2_ID	
2	HEX	000A	XSF_PSF2_LENGTH	
1	HEX	01	XSF_PSF2_DTYPE	
8	HEX	001087D000000007	XSF_SFI_PSN	Pool definition system ID
3	HEX	87D000	XSF_SFI_PSN_ID	
2	HEX	0010	XSF_PSN_LENGTH	
1	HEX	07	XSF_PSN_DTYPE	
8	HEX	001687D300000008	XSF_SFI_PSZ6	Physical space used
3	HEX	87D300	XSF_SFI_PSZ6_ID	
2	HEX	0016	XSF_PSZ6_LENGTH	
1	HEX	08	XSF_PSZ6_DTYPE	Compound data
6	HEX	010303010A06	XSF_SFI_PSZ6_COMP	
1	HEX	01	XSF_SFI_PSZ6_COMPTYPE	
1	HEX	03	XSF_SFI_PSZ6_COMPLEN1	
1	HEX	03	XSF_SFI_PSZ6_COMPTYP1	
1	HEX	01	XSF_SFI_PSZ6_COMPDAT1	
1	HEX	0A	XSF_SFI_PSZ6_COMPLEN2	
1	HEX	06	XSF_SFI_PSZ6_COMPTYP2	
8	HEX	000987DB00000003	XSF_SFI_PTNA	PRTITION action for non-system-managed volumes
3	HEX	87DB00	XSF_SFI_PTNA_ID	
2	HEX	0009	XSF_PTNA_LENGTH	
1	HEX	03	XSF_PTNA_DTYPE	
1	NUMB HEX	00	XSF_PTNA_DATA_ACCEPT	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	01	XSF_PTNA_DATA_IGNORE	
8	HEX	001087DB0C000007	XSF_SFI_PTNL	PRTITION location name
3	HEX	87DB0C	XSF_SFI_PTNL_ID	
2	HEX	0010	XSF_PTNL_LENGTH	
1	HEX	07	XSF_PTNL_DTYPE	
8	HEX	000987E000000003	XSF_SFI_PTP	Pool definition pool type
3	HEX	87E000	XSF_SFI_PTP_ID	
2	HEX	0009	XSF_PTP_LENGTH	
1	HEX	03	XSF_PTP_DTYPE	
1	NUMB HEX	00	XSF_PTP_DATA_SCRATCH	
1	NUMB HEX	01	XSF_PTP_DATA_RACK	
8	HEX	000987EB80000003	XSF_SFI_PTSA	PRTITION action for system-managed volumes
3	HEX	87EB80	XSF_SFI_PTSA_ID	
2	HEX	0009	XSF_PTSA_LENGTH	
1	HEX	03	XSF_PTSA_DTYPE	
1	NUMB HEX	00	XSF_PTSA_DATA_ACCEPT	
1	NUMB HEX	01	XSF_PTSA_DATA_IGNORE	
8	HEX	000987EBA8000003	XSF_SFI_PTTP	PRTITION entry type
3	HEX	87EBA8	XSF_SFI_PTTP_ID	
2	HEX	0009	XSF_PTTP_LENGTH	
1	HEX	03	XSF_PTTP_DTYPE	
1	NUMB HEX	00	XSF_PTTP_DATA_RMM	
1	NUMB HEX	01	XSF_PTTP_DATA_NORMM	
8	HEX	000E87EC00000007	XSF_SFI_PTVS	PRTITION volume range start
3	HEX	87EC00	XSF_SFI_PTVS_ID	
2	HEX	000E	XSF_PTVS_LENGTH	
1	HEX	07	XSF_PTVS_DTYPE	
8	HEX	000E87EC08000007	XSF_SFI_PTVL	PRTITION volume serial number, specific or generic
3	HEX	87EC08	XSF_SFI_PTVL_ID	
2	HEX	000E	XSF_PTVL_LENGTH	
1	HEX	07	XSF_PTVL_DTYPE	
8	HEX	000E87EC0F000007	XSF_SFI_PTVE	PRTITION volume range end
3	HEX	87EC0F	XSF_SFI_PTVE_ID	
2	HEX	000E	XSF_PTVE_LENGTH	
1	HEX	07	XSF_PTVE_DTYPE	
8	HEX	000E87F000000001	XSF_SFI_PVL	Previous volume
3	HEX	87F000	XSF_SFI_PVL_ID	
2	HEX	000E	XSF_PVL_LENGTH	
1	HEX	01	XSF_PVL_DTYPE	
8	HEX	000C880000000005	XSF_SFI_PWT	Permanent write errors
3	HEX	880000	XSF_SFI_PWT_ID	
2	HEX	000C	XSF_PWT_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	05	XSF_PWT_DTYPE	
8	HEX	000C881000000005	XSF_SFI_RBN	Number of bin numbers in REMOTE
3	HEX	881000	XSF_SFI_RBN_ID	
2	HEX	000C	XSF_RBN_LENGTH	
1	HEX	05	XSF_RBN_DTYPE	
8	HEX	0009881200000003	XSF_SFI_RBYS	Retain by set
3	HEX	881200	XSF_SFI_RBYS_ID	
2	HEX	0009	XSF_RBYS_LENGTH	
1	HEX	03	XSF_RBYS_DTYPE	
1	NUMB HEX	00	XSF_RBYS_DATA_NO	
1	NUMB HEX	01	XSF_RBYS_DATA_YES	
8	HEX	0009882000000003	XSF_SFI_RCF	Installation RACF support
3	HEX	882000	XSF_SFI_RCF_ID	
2	HEX	0009	XSF_RCF_LENGTH	
1	HEX	03	XSF_RCF_DTYPE	
1	NUMB HEX	01	XSF_RCF_DATA_N	
1	NUMB HEX	02	XSF_RCF_DATA_P	
1	NUMB HEX	03	XSF_RCF_DATA_A	
8	HEX	000C883000000007	XSF_SFI_RCFM	Record format
3	HEX	883000	XSF_SFI_RCFM_ID	
2	HEX	000C	XSF_RCFM_LENGTH	
1	HEX	07	XSF_RCFM_DTYPE	
8	HEX	000E884000000001	XSF_SFI_RCK	Rack or bin number
3	HEX	884000	XSF_SFI_RCK_ID	
2	HEX	000E	XSF_RCK_LENGTH	
1	HEX	01	XSF_RCK_DTYPE	
8	HEX	000C886000000009	XSF_SFI_RDTJ	Last CDS extract date
3	HEX	886000	XSF_SFI_RDTJ_ID	
2	HEX	000C	XSF_RDTJ_LENGTH	
1	HEX	09	XSF_RDTJ_DTYPE	
8	HEX	000B888000000003	XSF_SFI_RET	Retention type
3	HEX	888000	XSF_SFI_RET_ID	
2	HEX	000B	XSF_RET_LENGTH	
1	HEX	03	XSF_RET_DTYPE	
1	NUMB HEX	01	XSF_RET_DATA1_WHILEC	
1	NUMB HEX	01	XSF_RET_DATA2_UNTILEXP	
1	NUMB HEX	01	XSF_RET_DATA3_CYCLES	
1	NUMB HEX	02	XSF_RET_DATA3_DAYS	
1	NUMB HEX	03	XSF_RET_DATA3_REFDAYS	
1	NUMB HEX	04	XSF_RET_DATA3_VOLUMES	
1	NUMB HEX	05	XSF_RET_DATA3_EXTDAYS	
1	NUMB HEX	06	XSF_RET_DATA3_DAYCYCL	
8	HEX	000C888500000005	XSF_SFI_RLPR	Required location priority

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	888500	XSF_SFI_RLPR_ID	
2	HEX	000C	XSF_RLPR_LENGTH	
1	HEX	05	XSF_RLPR_DTYPE	
8	HEX	0009888800000003	XSF_SFI_RM	Retention method
3	HEX	888800	XSF_SFI_RM_ID	
2	HEX	0009	XSF_RM_LENGTH	
1	HEX	03	XSF_RM_DTYPE	
1	NUMB HEX	00	XSF_RM_DATA_VRSEL	
1	NUMB HEX	01	XSF_RM_DATA_EXPDT	
8	HEX	0009888A00000003	XSF_SFI_RMSB	Retention method set by
3	HEX	888A00	XSF_SFI_RMSB_ID	
2	HEX	0009	XSF_RMSB_LENGTH	
1	HEX	03	XSF_RMSB_DTYPE	
1	NUMB HEX	00	XSF_RMSB_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_RMSB_DATA_CMD	
1	NUMB HEX	02	XSF_RMSB_DATA_CMD_DEF	
1	NUMB HEX	03	XSF_RMSB_DATA_OCE_DEF	
1	NUMB HEX	04	XSF_RMSB_DATA_OCE_EXIT	
1	NUMB HEX	05	XSF_RMSB_DATA_LCS_DEF	
1	NUMB HEX	06	XSF_RMSB_DATA_CNVT	
1	NUMB HEX	07	XSF_RMSB_DATA_EXPORT_DEF	
1	NUMB HEX	08	XSF_RMSB_DATA_INERS_DEF	
1	NUMB HEX	09	XSF_RMSB_DATA_MC_ATTR	
1	NUMB HEX	0A	XSF_RMSB_DATA_DEFTABLE	
8	HEX	0019889000000007	XSF_SFI_RMID	RMM started procedure name
3	HEX	889000	XSF_SFI_RMID_ID	
2	HEX	0019	XSF_RMID_LENGTH	
1	HEX	07	XSF_RMID_DTYPE	
8	HEX	000988A000000003	XSF_SFI_RST	Rack or bin status
3	HEX	88A000	XSF_SFI_RST_ID	
2	HEX	0009	XSF_RST_LENGTH	
1	HEX	03	XSF_RST_DTYPE	
1	NUMB HEX	00	XSF_RST_DATA_EMPTY	
1	NUMB HEX	01	XSF_RST_DATA_FREE	
1	NUMB HEX	02	XSF_RST_DATA_INUSE	
8	HEX	000988B900000003	XSF_SFI_RTBY	Retain by
3	HEX	88B900	XSF_SFI_RTBY_ID	
2	HEX	0009	XSF_RTBY_LENGTH	
1	HEX	03	XSF_RTBY_DTYPE	
1	NUMB HEX	00	XSF_RTBY_DATA_VOLUME	
1	NUMB HEX	01	XSF_RTBY_DATA_SET	
8	HEX	000C88C000000009	XSF_SFI_RTDJ	Retention date
3	HEX	88C000	XSF_SFI_RTDJ_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	000C	XSF_RTDJ_LENGTH	
1	HEX	09	XSF_RTDJ_DTYPE	
8	HEX	000C88E00000000A	XSF_SFI_RTM	Last CDS extract time
3	HEX	88E000	XSF_SFI_RTM_ID	
2	HEX	000C	XSF_RTM_LENGTH	
1	HEX	0A	XSF_RTM_DTYPE	
8	HEX	000988E500000003	XSF_SFI_RUB	Reuse bin
3	HEX	88E500	XSF_SFI_RUB_ID	
2	HEX	0009	XSF_RUB_LENGTH	
1	HEX	03	XSF_RUB_DTYPE	
1	NUMB HEX	00	XSF_RUB_DATA_CONFIRMMOVE	
1	NUMB HEX	01	XSF_RUB_DATA_STARTMOVE	
8	HEX	0010890000000007	XSF_SFI_SC	Storage class
3	HEX	890000	XSF_SFI_SC_ID	
2	HEX	0010	XSF_SC_LENGTH	
1	HEX	07	XSF_SC_DTYPE	
8	HEX	0009891000000003	XSF_SFI_SCRM	Scratch mode
3	HEX	891000	XSF_SFI_SCRM_ID	
2	HEX	0009	XSF_SCRM_LENGTH	
1	HEX	03	XSF_SCRM_DTYPE	
8	HEX	0009892000000002	XSF_SFI_SCST	Security class status
3	HEX	892000	XSF_SFI_SCST_ID	
2	HEX	0009	XSF_SCST_LENGTH	
1	HEX	02	XSF_SCST_DTYPE	
1	HEX	80	XSF_SCST_FLAG_SMF	
1	HEX	40	XSF_SCST_FLAG_MSGOPT	
1	HEX	20	XSF_SCST_FLAG_ERASE	
8	HEX	000C894000000005	XSF_SFI_SC1	Store number
3	HEX	894000	XSF_SFI_SC1_ID	
2	HEX	000C	XSF_SC1_LENGTH	
1	HEX	05	XSF_SC1_DTYPE	
8	HEX	000C895000000009	XSF_SFI_SDTJ	Movement tracking date
3	HEX	895000	XSF_SFI_SDTJ_ID	
2	HEX	000C	XSF_SDTJ_LENGTH	
1	HEX	09	XSF_SDTJ_DTYPE	
8	HEX	0009896000000003	XSF_SFI_SEC	Security class number
3	HEX	896000	XSF_SFI_SEC_ID	
2	HEX	0009	XSF_SEC_LENGTH	
1	HEX	03	XSF_SEC_DTYPE	
8	HEX	000C898000000005	XSF_SFI_SEQ	Volume sequence
3	HEX	898000	XSF_SFI_SEQ_ID	
2	HEX	000C	XSF_SEQ_LENGTH	
1	HEX	05	XSF_SEQ_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	001089A000000007	XSF_SFI_SG	Storage group name
3	HEX	89A000	XSF_SFI_SG_ID	
2	HEX	0010	XSF_SG_LENGTH	
1	HEX	07	XSF_SG_DTYPE	
8	HEX	001089B000000007	XSF_SFI_SID	SMF system ID
3	HEX	89B000	XSF_SFI_SID_ID	
2	HEX	0010	XSF_SID_LENGTH	
1	HEX	07	XSF_SID_DTYPE	
8	HEX	000A89C000000004	XSF_SFI_SLM	MAXHOLD value
3	HEX	89C000	XSF_SFI_SLM_ID	
2	HEX	000A	XSF_SLM_LENGTH	
1	HEX	04	XSF_SLM_DTYPE	
8	HEX	000A89E000000004	XSF_SFI_SMI	Offset to message ID
3	HEX	89E000	XSF_SFI_SMI_ID	
2	HEX	000A	XSF_SMI_LENGTH	
1	HEX	04	XSF_SMI_DTYPE	
8	HEX	000989E210000003	XSF_SFI_SMP	SMSTAPE purge
3	HEX	89E210	XSF_SFI_SMP_ID	
2	HEX	0009	XSF_SMP_LENGTH	
1	HEX	03	XSF_SMP_DTYPE	
8	HEX	000989E220000002	XSF_SFI_SMU	SMSTAPE update
3	HEX	89E220	XSF_SFI_SMU_ID	
2	HEX	0009	XSF_SMU_LENGTH	
1	HEX	02	XSF_SMU_DTYPE	
8	HEX	000C89F000000009	XSF_SFI_SOSJ	Last expiration processing start date
3	HEX	89F000	XSF_SFI_SOSJ_ID	
2	HEX	000C	XSF_SOSJ_LENGTH	
1	HEX	09	XSF_SOSJ_DTYPE	
8	HEX	00108A0000000007	XSF_SFI_SOSP	Scratch procedure name
3	HEX	8A0000	XSF_SFI_SOSP_ID	
2	HEX	0010	XSF_SOSP_LENGTH	
1	HEX	07	XSF_SOSP_DTYPE	
8	HEX	000C8A100000000A	XSF_SFI_SOST	Last expiration processing start time
3	HEX	8A1000	XSF_SFI_SOST_ID	
2	HEX	000C	XSF_SOST_LENGTH	
1	HEX	0A	XSF_SOST_DTYPE	
8	HEX	00478A1A00000007	XSF_SFI_SRHN	Server host name
3	HEX	8A1A00	XSF_SFI_SRHN_ID	
2	HEX	0047	XSF_SRHN_LENGTH	
1	HEX	07	XSF_SRHN_DTYPE	
8	HEX	00358A1A30000007	XSF_SFI_SRIP	Server IP address
3	HEX	8A1A30	XSF_SFI_SRIP_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	0035	XSF_SRIP_LENGTH	
1	HEX	07	XSF_SRIP_DTYPE	
8	HEX	000C8A1A50000005	XSF_SFI_SRPN	Server port number
3	HEX	8A1A50	XSF_SFI_SRPN_ID	
2	HEX	000C	XSF_SRPN_LENGTH	
1	HEX	05	XSF_SRPN_DTYPE	
8	HEX	000C8A1AF0000005	XSF_SFI_SRTK	Server tasks
3	HEX	8A1AF0	XSF_SFI_SRTK_ID	
2	HEX	000C	XSF_SRTK_LENGTH	
1	HEX	05	XSF_SRTK_DTYPE	
8	HEX	000A8A2000000004	XSF_SFI_SSM	SMF security record number
3	HEX	8A2000	XSF_SFI_SSM_ID	
2	HEX	000A	XSF_SSM_LENGTH	
1	HEX	04	XSF_SSM_DTYPE	
8	HEX	00098A2500000003	XSF_SFI_SSTY	Subsystem type
3	HEX	8A2500	XSF_SFI_SSTY_ID	
2	HEX	0009	XSF_SSTY_LENGTH	
1	HEX	03	XSF_SSTY_DTYPE	
1	NUMB HEX	00	XSF_SSTY_DATA_STANDARD	
1	NUMB HEX	01	XSF_SSTY_DATA_CLIENT	
1	NUMB HEX	02	XSF_SSTY_DATA_SERVER	
8	HEX	00108A3000000007	XSF_SFI_STEP	Step name
3	HEX	8A3000	XSF_SFI_STEP_ID	
2	HEX	0010	XSF_STEP_LENGTH	
1	HEX	07	XSF_STEP_DTYPE	
8	HEX	00098A2800000002	XSF_SFI_STDS	Debug setting
3	HEX	8A2800	XSF_SFI_STDS_ID	
2	HEX	0009	XSF_STDS_LENGTH	
1	HEX	02	XSF_STDS_DTYPE	
1	HEX	80	XSF_STDS_FLAG_OCE	
1	HEX	40	XSF_STDS_FLAG_SNAP	
8	HEX	00098A3607000003	XSF_SFI_STRM	DFSMSrmm status
3	HEX	8A3607	XSF_SFI_STRM_ID	
2	HEX	0009	XSF_STRM_LENGTH	
1	HEX	03	XSF_STRM_DTYPE	
1	NUMB HEX	00	XSF_STRM_DATA_ACTIVE	
1	NUMB HEX	01	XSF_STRM_DATA_RESET	
1	NUMB HEX	02	XSF_STRM_DATA_QUIESCED	
8	HEX	00098A3661000003	XSF_SFI_STSL	Server listener status
3	HEX	8A3661	XSF_SFI_STSL_ID	
2	HEX	0009	XSF_STSL_LENGTH	
1	HEX	03	XSF_STSL_DTYPE	
1	NUMB HEX	00	XSF_STSL_DATA_STANDARD	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	01	XSF_STSL_DATA_ACTIVE	
1	NUMB HEX	02	XSF_STSL_DATA_INACTIVE	
8	HEX	000A8A3314000004	XSF_SFI_STLO	Local tasks
3	HEX	8A3314	XSF_SFI_STLO_ID	
2	HEX	000A	XSF_STLO_LENGTH	
1	HEX	04	XSF_STLO_DTYPE	
8	HEX	000A8A3300000004	XSF_SFI_STLA	Local active tasks
3	HEX	8A3300	XSF_SFI_STLA_ID	
2	HEX	000A	XSF_STLA_LENGTH	
1	HEX	04	XSF_STLA_DTYPE	
8	HEX	000A8A3307000004	XSF_SFI_STLH	Local held tasks
3	HEX	8A3307	XSF_SFI_STLH_ID	
2	HEX	000A	XSF_STLH_LENGTH	
1	HEX	04	XSF_STLH_DTYPE	
8	HEX	000A8A3664000004	XSF_SFI_STSO	Server tasks
3	HEX	8A3664	XSF_SFI_STSO_ID	
2	HEX	000A	XSF_STSO_LENGTH	
1	HEX	04	XSF_STSO_DTYPE	
8	HEX	000A8A3650000004	XSF_SFI_STSA	Server active tasks
3	HEX	8A3650	XSF_SFI_STSA_ID	
2	HEX	000A	XSF_STSA_LENGTH	
1	HEX	04	XSF_STSA_DTYPE	
8	HEX	000A8A3657000004	XSF_SFI_STSH	Server held tasks
3	HEX	8A3657	XSF_SFI_STSH_ID	
2	HEX	000A	XSF_STSH_LENGTH	
1	HEX	04	XSF_STSH_DTYPE	
8	HEX	000C8A3515000005	XSF_SFI_STQR	Queued requests
3	HEX	8A3515	XSF_SFI_STQR_ID	
2	HEX	000C	XSF_STQR_LENGTH	
1	HEX	05	XSF_STQR_DTYPE	
8	HEX	000C8A3511000005	XSF_SFI_STQN	Queued nowait requests
3	HEX	8A3511	XSF_SFI_STQN_ID	
2	HEX	000C	XSF_STQN_LENGTH	
1	HEX	05	XSF_STQN_DTYPE	
8	HEX	000C8A3500000005	XSF_SFI_STQC	Queued catalog requests
3	HEX	8A3500	XSF_SFI_STQC_ID	
2	HEX	000C	XSF_STQC_LENGTH	
1	HEX	05	XSF_STQC_DTYPE	
8	HEX	000C8A331700000A	XSF_SFI_STLR	Last reserve time
3	HEX	8A3317	XSF_SFI_STLR_ID	
2	HEX	000C	XSF_STLR_LENGTH	
1	HEX	0A	XSF_STLR_DTYPE	
8	HEX	00098A3400000003	XSF_SFI_STNH	New requests held

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	8A3400	XSF_SFI_STNH_ID	
2	HEX	0009	XSF_STNH_LENGTH	
1	HEX	03	XSF_STNH_DTYPE	
1	NUMB HEX	00	XSF_STNH_DATA_NOTHELD	
1	NUMB HEX	01	XSF_STNH_DATA_HELD	
8	HEX	00098A3602000003	XSF_SFI_STRH	CDS reserved
3	HEX	8A3602	XSF_SFI_STRH_ID	
2	HEX	0009	XSF_STRH_LENGTH	
1	HEX	03	XSF_STRH_DTYPE	
1	NUMB HEX	00	XSF_STRH_DATA_DEQ	
1	NUMB HEX	01	XSF_STRH_DATA_ENQ	
8	HEX	00098A3450000002	XSF_SFI_STPL	Trace levels
3	HEX	8A3450	XSF_SFI_STPL_ID	
2	HEX	0009	XSF_STPL_LENGTH	
1	HEX	02	XSF_STPL_DTYPE	
1	HEX	80	XSF_STPL_DATA_LVL1	
1	HEX	40	XSF_STPL_DATA_LVL2	
1	HEX	20	XSF_STPL_DATA_LVL3	
1	HEX	10	XSF_STPL_DATA_LVL4	
8	HEX	000D8A3600000007	XSF_SFI_STRF	Task requesting function
3	HEX	8A3600	XSF_SFI_STRF_ID	
2	HEX	000D	XSF_STRF_LENGTH	
1	HEX	07	XSF_STRF_DTYPE	
8	HEX	00108A3614000007	XSF_SFI_STRT	Task requesting system
3	HEX	8A3614	XSF_SFI_STRT_ID	
2	HEX	0010	XSF_STRT_LENGTH	
1	HEX	07	XSF_STRT_DTYPE	
8	HEX	000B8A3701000007	XSF_SFI_STTR	Task requesting type
3	HEX	8A3701	XSF_SFI_STTR_ID	
2	HEX	000B	XSF_STTR_LENGTH	
1	HEX	07	XSF_STTR_DTYPE	
8	HEX	00108A3700000007	XSF_SFI_STTQ	Task requestor
3	HEX	8A3700	XSF_SFI_STTQ_ID	
2	HEX	0010	XSF_STTQ_LENGTH	
1	HEX	07	XSF_STTQ_DTYPE	
8	HEX	000C8A366900000A	XSF_SFI_STST	Task start time
3	HEX	8A3669	XSF_SFI_STST_ID	
2	HEX	000C	XSF_STST_LENGTH	
1	HEX	0A	XSF_STST_DTYPE	
8	HEX	000C8A3703000005	XSF_SFI_STTT	Task token
3	HEX	8A3703	XSF_SFI_STTT_ID	
2	HEX	000C	XSF_STTT_LENGTH	
1	HEX	05	XSF_STTT_DTYPE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
8	HEX	00098A3702000003	XSF_SFI_STTS	Task status
3	HEX	8A3702	XSF_SFI_STTS_ID	
2	HEX	0009	XSF_STTS_LENGTH	
1	HEX	03	XSF_STTS_DTYPE	
1	NUMB HEX	00	XSF_STTS_DATA_NONE	
1	NUMB HEX	01	XSF_STTS_DATA_HOLD	
1	NUMB HEX	02	XSF_STTS_DATA_CANCEL	
1	NUMB HEX	03	XSF_STTS_DATA_RESERVE	
8	HEX	00098A3203000003	XSF_SFI_STIV	IP verb
3	HEX	8A3203	XSF_SFI_STIV_ID	
2	HEX	0009	XSF_STIV_LENGTH	
1	HEX	03	XSF_STIV_DTYPE	
1	NUMB HEX	00	XSF_STIV_DATA_NONE	
1	NUMB HEX	01	XSF_STIV_DATA_READ	
1	NUMB HEX	02	XSF_STIV_DATA_WRITE	
1	NUMB HEX	03	XSF_STIV_DATA_CONNECT	
1	NUMB HEX	04	XSF_STIV_DATA_CLOSE	
8	HEX	00098A3200000003	XSF_SFI_STIS	IP verb state
3	HEX	8A3200	XSF_SFI_STIS_ID	
2	HEX	0009	XSF_STIS_LENGTH	
1	HEX	03	XSF_STIS_DTYPE	
1	NUMB HEX	00	XSF_STIS_DATA_NONE	
1	NUMB HEX	01	XSF_STIS_DATA_STARTED	
1	NUMB HEX	02	XSF_STIS_DATA_ENDED	
8	HEX	000C8A320100000A	XSF_SFI_STIT	IP verb time
3	HEX	8A3201	XSF_SFI_STIT_ID	
2	HEX	000C	XSF_STIT_LENGTH	
1	HEX	0A	XSF_STIT_DTYPE	
8	HEX	000C8A3800000005	XSF_SFI_STVC	Stacked volume count
3	HEX	8A3800	XSF_SFI_STVC_ID	
2	HEX	000C	XSF_STVC_LENGTH	
1	HEX	05	XSF_STVC_DTYPE	
8	HEX	001C8A4000000007	XSF_SFI_SUR	Owner's surname
3	HEX	8A4000	XSF_SFI_SUR_ID	
2	HEX	001C	XSF_SUR_LENGTH	
1	HEX	07	XSF_SUR_DTYPE	
8	HEX	00108A5000000007	XSF_SFI_SYS	Creating system ID
3	HEX	8A5000	XSF_SFI_SYS_ID	
2	HEX	0010	XSF_SYS_LENGTH	
1	HEX	07	XSF_SYS_DTYPE	
8	HEX	00098A6000000003	XSF_SFI_TAC	Reject type
3	HEX	8A6000	XSF_SFI_TAC_ID	
2	HEX	0009	XSF_TAC_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	03	XSF_TAC_DTYPE	
1	NUMB HEX	00	XSF_TAC_DATA_ANYUSE	
1	NUMB HEX	01	XSF_TAC_DATA_OUTPUT	
8	HEX	000C8A680000000A	XSF_SFI_TLR	Time last referenced
3	HEX	8A6800	XSF_SFI_TLR_ID	
2	HEX	000C	XSF_TLR_LENGTH	
1	HEX	0A	XSF_TLR_DTYPE	
8	HEX	000C8A7000000005	XSF_SFI_TRD	Temporary read errors
3	HEX	8A7000	XSF_SFI_TRD_ID	
2	HEX	000C	XSF_TRD_LENGTH	
1	HEX	05	XSF_TRD_DTYPE	
8	HEX	000C8A7800000005	XSF_SFI_TVXD	TVEXTPURGE(EXPIRE(days))
3	HEX	8A7800	XSF_SFI_TVXD_ID	
2	HEX	000C	XSF_TVXD_LENGTH	
1	HEX	05	XSF_TVXD_DTYPE	
8	HEX	00098A7900000003	XSF_SFI_TVXP	Tape volume exit purge
3	HEX	8A7900	XSF_SFI_TVXP_ID	
2	HEX	0009	XSF_TVXP_LENGTH	
1	HEX	03	XSF_TVXP_DTYPE	
8	HEX	000C8A8000000005	XSF_SFI_TWT	Temporary write errors
3	HEX	8A8000	XSF_SFI_TWT_ID	
2	HEX	000C	XSF_TWT_LENGTH	
1	HEX	05	XSF_TWT_DTYPE	
8	HEX	00098A9000000002	XSF_SFI_TYP	VRS type
3	HEX	8A9000	XSF_SFI_TYP_ID	
2	HEX	0009	XSF_TYP_LENGTH	
1	HEX	02	XSF_TYP_DTYPE	
1	HEX	80	XSF_TYP_FLAG_GDG	
1	HEX	40	XSF_TYP_FLAG_PSEUDGDG	
1	HEX	20	XSF_TYP_FLAG_DSNAME	
1	HEX	10	XSF_TYP_FLAG_VOLUME	
1	HEX	08	XSF_TYP_FLAG_NAME	
8	HEX	000C8A9E00000005	XSF_SFI_TZ	Time zone
3	HEX	8A9E00	XSF_SFI_TZ_ID	
2	HEX	000C	XSF_TZ_LENGTH	
1	HEX	05	XSF_TZ_DTYPE	
8	HEX	000C8AA000000009	XSF_SFI_UDTJ	Last update date
3	HEX	8AA000	XSF_SFI_UDTJ_ID	
2	HEX	000C	XSF_UDTJ_LENGTH	
1	HEX	09	XSF_UDTJ_DTYPE	
8	HEX	00108AB001000007	XSF_SFI_UID	User ID
3	HEX	8AB001	XSF_SFI_UID_ID	
2	HEX	0010	XSF_UID_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	07	XSF_UID_DTYPE	
8	HEX	00098AC000000003	XSF_SFI_UNC	Uncatalog option
3	HEX	8AC000	XSF_SFI_UNC_ID	
2	HEX	0009	XSF_UNC_LENGTH	
1	HEX	03	XSF_UNC_DTYPE	
1	NUMB HEX	00	XSF_UNC_DATA_N	
1	NUMB HEX	01	XSF_UNC_DATA_Y	
1	NUMB HEX	02	XSF_UNC_DATA_S	
8	HEX	000C8AD000000005	XSF_SFI_USEC	Volume use count
3	HEX	8AD000	XSF_SFI_USEC_ID	
2	HEX	000C	XSF_USEC_LENGTH	
1	HEX	05	XSF_USEC_DTYPE	
8	HEX	000C8AE000000005	XSF_SFI_USEM	Volume usage (KB)
3	HEX	8AE000	XSF_SFI_USEM_ID	
2	HEX	000C	XSF_USEM_LENGTH	
1	HEX	05	XSF_USEM_DTYPE	
8	HEX	00168AE030000008	XSF_SFI_USE6	Volume usage
3	HEX	8AE030	XSF_SFI_USE6_ID	
2	HEX	0016	XSF_USE6_LENGTH	
1	HEX	08	XSF_USE6_DTYPE	
6	HEX	010303010A06	XSF_SFI_USE6_COMP	Compound data
1	HEX	01	XSF_SFI_USE6_COMPTYPE	
1	HEX	03	XSF_SFI_USE6_COMPLEN1	
1	HEX	03	XSF_SFI_USE6_COMPTYP1	
1	HEX	01	XSF_SFI_USE6_COMPDAT1	
1	HEX	0A	XSF_SFI_USE6_COMPLEN2	
1	HEX	06	XSF_SFI_USE6_COMPTYP2	
8	HEX	00098AE600000003	XSF_SFI_UTC	Common time enabled status
3	HEX	8AE600	XSF_SFI_UTC_ID	
2	HEX	0009	XSF_UTC_LENGTH	
1	HEX	03	XSF_UTC_DTYPE	
1	NUMB HEX	00	XSF_UTC_DATA_DISABLED	
1	NUMB HEX	01	XSF_UTC_DATA_ENABLED	
8	HEX	000C8AE80000000A	XSF_SFI_UTM	Last update time
3	HEX	8AE800	XSF_SFI_UTM_ID	
2	HEX	000C	XSF_UTM_LENGTH	
1	HEX	0A	XSF_UTM_DTYPE	
8	HEX	00098AF001000003	XSF_SFI_VAC	Volume access
3	HEX	8AF001	XSF_SFI_VAC_ID	
2	HEX	0009	XSF_VAC_LENGTH	
1	HEX	03	XSF_VAC_DTYPE	
1	NUMB HEX	00	XSF_VAC_DATA_NONE	
1	NUMB HEX	01	XSF_VAC_DATA_READ	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	NUMB HEX	02	XSF_VAC_DATA_UPDATE	
8	HEX	00098B0000000003	XSF_SFI_VACT	VRSMIN action
3	HEX	8B0000	XSF_SFI_VACT_ID	
2	HEX	0009	XSF_VACT_LENGTH	
1	HEX	03	XSF_VACT_DTYPE	
1	NUMB HEX	00	XSF_VACT_DATA_FAIL	
1	NUMB HEX	01	XSF_VACT_DATA_INFO	
1	NUMB HEX	02	XSF_VACT_DATA_WARN	
1	NUMB HEX	03	XSF_VACT_DATA_OFF	
8	HEX	00098B2800000003	XSF_SFI_VDRA	VRSDROP action
3	HEX	8B2800	XSF_SFI_VDRA_ID	
2	HEX	0009	XSF_VDRA_LENGTH	
1	HEX	03	XSF_VDRA_DTYPE	
1	NUMB HEX	00	XSF_VDRA_DATA_FAIL	
1	NUMB HEX	01	XSF_VDRA_DATA_INFO	
1	NUMB HEX	02	XSF_VDRA_DATA_WARN	
1	NUMB HEX	03	XSF_VDRA_DATA_OFF	
8	HEX	00098BD500000003	XSF_SFI_VREA	VRSRETAIN action
3	HEX	8BD500	XSF_SFI_VREA_ID	
2	HEX	0009	XSF_VREA_LENGTH	
1	HEX	03	XSF_VREA_DTYPE	
1	NUMB HEX	00	XSF_VREA_DATA_FAIL	
1	NUMB HEX	01	XSF_VREA_DATA_INFO	
1	NUMB HEX	02	XSF_VREA_DATA_WARN	
1	NUMB HEX	03	XSF_VREA_DATA_OFF	
8	HEX	00098C5D00000003	XSF_SFI_XDRA	EXPDTDROP action
3	HEX	8C5D00	XSF_SFI_XDRA_ID	
2	HEX	0009	XSF_XDRA_LENGTH	
1	HEX	03	XSF_XDRA_DTYPE	
1	NUMB HEX	00	XSF_XDRA_DATA_FAIL	
1	NUMB HEX	01	XSF_XDRA_DATA_INFO	
1	NUMB HEX	02	XSF_XDRA_DATA_WARN	
1	NUMB HEX	03	XSF_XDRA_DATA_OFF	
8	HEX	00098B0800000003	XSF_SFI_VANX	Next VRS type
3	HEX	8B0800	XSF_SFI_VANX_ID	
2	HEX	0009	XSF_VANX_LENGTH	
1	HEX	03	XSF_VANX_DTYPE	
1	NUMB HEX	00	XSF_VANX_DATA_NONE	
1	NUMB HEX	01	XSF_VANX_DATA_NEXT	
1	NUMB HEX	02	XSF_VANX_DATA_AND	
8	HEX	000C8B0B00000005	XSF_SFI_VCAP	Volume capacity
3	HEX	8B0B00	XSF_SFI_VCAP_ID	
2	HEX	000C	XSF_VCAP_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	05	XSF_VCAP_DTYPE	
8	HEX	00098B1000000003	XSF_SFI_VCHG	VRSCHANGE value
3	HEX	8B1000	XSF_SFI_VCHG_ID	
2	HEX	0009	XSF_VCHG_LENGTH	
1	HEX	03	XSF_VCHG_DTYPE	
1	NUMB HEX	00	XSF_VCHG_DATA_INFO	
1	NUMB HEX	01	XSF_VCHG_DATA_VERIFY	
8	HEX	000A8B2000000004	XSF_SFI_VDD	VRS delay days
3	HEX	8B2000	XSF_SFI_VDD_ID	
2	HEX	000A	XSF_VDD_LENGTH	
1	HEX	04	XSF_VDD_DTYPE	
8	HEX	000C8B3000000009	XSF_SFI_VDTJ	Last inventory management processing date
3	HEX	8B3000	XSF_SFI_VDTJ_ID	
2	HEX	000C	XSF_VDTJ_LENGTH	
1	HEX	09	XSF_VDTJ_DTYPE	
8	HEX	000E8B4000000007	XSF_SFI_VER	Software product version
3	HEX	8B4000	XSF_SFI_VER_ID	
2	HEX	000E	XSF_VER_LENGTH	
1	HEX	07	XSF_VER_DTYPE	
8	HEX	00098B4100000003	XSF_SFI_VEX	VRSEL EXCLUDE
3	HEX	8B4100	XSF_SFI_VEX_ID	
2	HEX	0009	XSF_VEX_LENGTH	
1	HEX	03	XSF_VEX_DTYPE	
1	NUMB HEX	00	XSF_VEX_DATA_NO	
1	NUMB HEX	01	XSF_VEX_DATA_YES	
8	HEX	00108B5000000007	XSF_SFI_VJBN	Primary VRS job name
3	HEX	8B5000	XSF_SFI_VJBN_ID	
2	HEX	0010	XSF_VJBN_LENGTH	
1	HEX	07	XSF_VJBN_DTYPE	
8	HEX	000C8B6000000005	XSF_SFI_VLN	Number of volumes
3	HEX	8B6000	XSF_SFI_VLN_ID	
2	HEX	000C	XSF_VLN_LENGTH	
1	HEX	05	XSF_VLN_DTYPE	
8	HEX	00098B7000000003	XSF_SFI_VM	VM use
3	HEX	8B7000	XSF_SFI_VM_ID	
2	HEX	0009	XSF_VM_LENGTH	
1	HEX	03	XSF_VM_DTYPE	
1	NUMB HEX	00	XSF_VM_DATA_NO	
1	NUMB HEX	01	XSF_VM_DATA_YES	
8	HEX	000C8B8000000005	XSF_SFI_VMIN	VRSMIN count value
3	HEX	8B8000	XSF_SFI_VMIN_ID	
2	HEX	000C	XSF_VMIN_LENGTH	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	HEX	05	XSF_VMIN_DTYPE	
8	HEX	000C8B2802000005	XSF_SFI_VDRC	VRSDROP count value
3	HEX	8B2802	XSF_SFI_VDRC_ID	
2	HEX	000C	XSF_VDRC_LENGTH	
1	HEX	05	XSF_VDRC_DTYPE	
8	HEX	000C8BD502000005	XSF_SFI_VREC	VRSRETAIN count value
3	HEX	8BD502	XSF_SFI_VREC_ID	
2	HEX	000C	XSF_VREC_LENGTH	
1	HEX	05	XSF_VREC_DTYPE	
8	HEX	000C8C5D02000005	XSF_SFI_XDRC	EXPDTDROP count value
3	HEX	8C5D02	XSF_SFI_XDRC_ID	
2	HEX	000C	XSF_XDRC_LENGTH	
1	HEX	05	XSF_XDRC_DTYPE	
8	HEX	000A8B280F000004	XSF_SFI_VDRP	VRSDROP percent value
3	HEX	8B280F	XSF_SFI_VDRP_ID	
2	HEX	000A	XSF_VDRP_LENGTH	
1	HEX	04	XSF_VDRP_DTYPE	
8	HEX	000A8BD50F000004	XSF_SFI_VREP	VRSRETAIN percent value
3	HEX	8BD50F	XSF_SFI_VREP_ID	
2	HEX	000A	XSF_VREP_LENGTH	
1	HEX	04	XSF_VREP_DTYPE	
8	HEX	000A8C5D0F000004	XSF_SFI_XDRP	EXPDTDROP percent value
3	HEX	8C5D0F	XSF_SFI_XDRP_ID	
2	HEX	000A	XSF_XDRP_LENGTH	
1	HEX	04	XSF_XDRP_DTYPE	
8	HEX	00108B9000000007	XSF_SFI_VMV	VRS management value
3	HEX	8B9000	XSF_SFI_VMV_ID	
2	HEX	0010	XSF_VMV_LENGTH	
1	HEX	07	XSF_VMV_DTYPE	
8	HEX	000C8B9100000005	XSF_SFI_VWMC	Volume write mount count
3	HEX	8B9100	XSF_SFI_VWMC_ID	
2	HEX	000C	XSF_VWMC_LENGTH	
1	HEX	05	XSF_VWMC_DTYPE	
8	HEX	00108B9E00000001	XSF_SFI_VNDR	Vendor
3	HEX	8B9E00	XSF_SFI_VNDR_ID	
2	HEX	0010	XSF_VNDR_LENGTH	
1	HEX	01	XSF_VNDR_DTYPE	
8	HEX	00348BA000000007	XSF_SFI_VNME	Primary VRS name
3	HEX	8BA000	XSF_SFI_VNME_ID	
2	HEX	0034	XSF_VNME_LENGTH	
1	HEX	07	XSF_VNME_DTYPE	
8	HEX	000E8BC000000001	XSF_SFI_VOL	Volume serial number
3	HEX	8BC000	XSF_SFI_VOL_ID	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
2	HEX	000E	XSF_VOL_LENGTH	
1	HEX	01	XSF_VOL_DTYPE	
8	HEX	00098BC200000003	XSF_SFI_VOLT	Volume type
3	HEX	8BC200	XSF_SFI_VOLT_ID	
2	HEX	0009	XSF_VOLT_LENGTH	
1	HEX	03	XSF_VOLT_DTYPE	
1	NUMB HEX	00	XSF_VOLT_PHYSICAL	
1	NUMB HEX	01	XSF_VOLT_LOGICAL	
1	NUMB HEX	02	XSF_VOLT_STACKED	
8	HEX	00098BC300000003	XSF_SFI_VPCT	Volume percent full
3	HEX	8BC300	XSF_SFI_VPCT_ID	
2	HEX	0009	XSF_VPCT_LENGTH	
1	HEX	03	XSF_VPCT_DTYPE	
8	HEX	000E8BCD00000001	XSF_SFI_VOL1	VOL1 label volume serial number
3	HEX	8BCD00	XSF_SFI_VOL1_ID	
2	HEX	000E	XSF_VOL1_LENGTH	
1	HEX	01	XSF_VOL1_DTYPE	
8	HEX	000C8BD000000005	XSF_SFI_VRC	Vital record count
3	HEX	8BD000	XSF_SFI_VRC_ID	
2	HEX	000C	XSF_VRC_LENGTH	
1	HEX	05	XSF_VRC_DTYPE	
8	HEX	00098BE000000003	XSF_SFI_VRJ	Vital record job name
3	HEX	8BE000	XSF_SFI_VRJ_ID	
2	HEX	0009	XSF_VRJ_LENGTH	
1	HEX	03	XSF_VRJ_DTYPE	
1	NUMB HEX	01	XSF_VRJ_DATA_1	
1	NUMB HEX	02	XSF_VRJ_DATA_2	
8	HEX	00348BF000000007	XSF_SFI_VRS	Vital record specification
3	HEX	8BF000	XSF_SFI_VRS_ID	
2	HEX	0034	XSF_VRS_LENGTH	
1	HEX	07	XSF_VRS_DTYPE	
8	HEX	00098BF500000003	XSF_SFI_VRSI	Scratch immediate
3	HEX	8BF500	XSF_SFI_VRSI_ID	
2	HEX	0009	XSF_VRSI_LENGTH	
1	HEX	03	XSF_VRSI_DTYPE	
1	NUMB HEX	00	XSF_VRSI_DATA_NO	
1	NUMB HEX	01	XSF_VRSI_DATA_YES	
8	HEX	00098BFA00000003	XSF_SFI_VRSL	VRSEL value
3	HEX	8BFA00	XSF_SFI_VRSL_ID	
2	HEX	0009	XSF_VRSL_LENGTH	
1	HEX	03	XSF_VRSL_DTYPE	
1	NUMB HEX	01	XSF_VRSL_DATA_NEW	
8	HEX	00098C0000000003	XSF_SFI_VRSR	VRS retained

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	8C0000	XSF_SFI_VRSR_ID	
2	HEX	0009	XSF_VRSR_LENGTH	
1	HEX	03	XSF_VRSR_DTYPE	
1	NUMB HEX	00	XSF_VRSR_DATA_NO	
1	NUMB HEX	01	XSF_VRSR_DATA_YES	
8	HEX	00098C0800000003	XSF_SFI_VRXI	Expiration date ignore
3	HEX	8C0800	XSF_SFI_VRXI_ID	
2	HEX	0009	XSF_VRXI_LENGTH	
1	HEX	03	XSF_VRXI_DTYPE	
1	NUMB HEX	00	XSF_VRXI_DATA_NO	
1	NUMB HEX	01	XSF_VRXI_DATA_YES	
8	HEX	000C8C1000000009	XSF_SFI_VSCD	Primary VRS subchain start date
3	HEX	8C1000	XSF_SFI_VSCD_ID	
2	HEX	000C	XSF_VSCD_LENGTH	
1	HEX	09	XSF_VSCD_DTYPE	
8	HEX	00108C1800000007	XSF_SFI_VSCN	Primary VRS subchain name
3	HEX	8C1800	XSF_SFI_VSCN_ID	
2	HEX	0010	XSF_VSCN_LENGTH	
1	HEX	07	XSF_VSCN_DTYPE	
8	HEX	00098C2000000002	XSF_SFI_VST	Volume status
3	HEX	8C2000	XSF_SFI_VST_ID	
2	HEX	0009	XSF_VST_LENGTH	
1	HEX	02	XSF_VST_DTYPE	
1	HEX	80	XSF_VST_FLAG_MASTER	
1	HEX	40	XSF_VST_FLAG_SCRATCH	
1	HEX	20	XSF_VST_FLAG_USER	
1	HEX	10	XSF_VST_FLAG_INIT	
1	HEX	08	XSF_VST_FLAG_ENTRY	
8	HEX	000C8C300000000A	XSF_SFI_VTM	Last inventory management VRS time
3	HEX	8C3000	XSF_SFI_VTM_ID	
2	HEX	000C	XSF_VTM_LENGTH	
1	HEX	0A	XSF_VTM_DTYPE	
8	HEX	00098C4000000003	XSF_SFI_VTYP	Primary VRS type
3	HEX	8C4000	XSF_SFI_VTYP_ID	
2	HEX	0009	XSF_VTYP_LENGTH	
1	HEX	03	XSF_VTYP_DTYPE	
1	NUMB HEX	00	XSF_VTYP_DATA_UNDEFINED	
1	NUMB HEX	01	XSF_VTYP_DATA_DATASET	
1	NUMB HEX	02	XSF_VTYP_DATA_SMSMC	
1	NUMB HEX	03	XSF_VTYP_DATA_VRSMV	
1	NUMB HEX	04	XSF_VTYP_DATA_DSNMV	
1	NUMB HEX	05	XSF_VTYP_DATA_DSNMC	
8	HEX	00098C4150000003	XSF_SFI_WCTL	Volume WHILECATALOG

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
3	HEX	8C4150	XSF_SFI_WCTL_ID	
2	HEX	0009	XSF_WCTL_LENGTH	
1	HEX	03	XSF_WCTL_DTYPE	
1	NUMB HEX	00	XSF_WCTL_DATA_OFF	
1	NUMB HEX	20	XSF_WCTL_DATA_ON	
1	NUMB HEX	10	XSF_WCTL_DATA_UNTILEXPIRED	
8	HEX	00098C4300000003	XSF_SFI_WORM	WORM flag
3	HEX	8C4300	XSF_SFI_WORM_ID	
2	HEX	0009	XSF_WORM_LENGTH	
1	HEX	03	XSF_WORM_DTYPE	
1	NUMB HEX	00	XSF_WORM_DATA_NO	
1	NUMB HEX	01	XSF_WORM_DATA_YES	
8	HEX	00208C4500000001	XSF_SFI_WWID	World wide ID
3	HEX	8C4500	XSF_SFI_WWID_ID	
2	HEX	0020	XSF_WWID_LENGTH	
1	HEX	01	XSF_WWID_DTYPE	
8	HEX	00098C5000000003	XSF_SFI_XDC	Expiration date check
3	HEX	8C5000	XSF_SFI_XDC_ID	
2	HEX	0009	XSF_XDC_LENGTH	
1	HEX	03	XSF_XDC_DTYPE	
1	NUMB HEX	00	XSF_XDC_DATA_NO	
1	NUMB HEX	01	XSF_XDC_DATA_YES	
1	NUMB HEX	02	XSF_XDC_DATA_OPERATOR	
8	HEX	000C8C6000000009	XSF_SFI_XDTJ	Expiration date
3	HEX	8C6000	XSF_SFI_XDTJ_ID	
2	HEX	000C	XSF_XDTJ_LENGTH	
1	HEX	09	XSF_XDTJ_DTYPE	
8	HEX	00098C6100000003	XSF_SFI_XDSB	Expiry date set by
3	HEX	8C6100	XSF_SFI_XDSB_ID	
2	HEX	0009	XSF_XDSB_LENGTH	
1	HEX	03	XSF_XDSB_DTYPE	
1	DECIMAL	0	XSF_XDSB_DATA_UNKNOWN	
1	DECIMAL	1	XSF_XDSB_DATA_CMD	
1	DECIMAL	2	XSF_XDSB_DATA_CMD_DEF	
1	DECIMAL	3	XSF_XDSB_DATA_CMD_VOLCAT	
1	DECIMAL	4	XSF_XDSB_DATA_OCE_JFCB	
1	DECIMAL	5	XSF_XDSB_DATA_OCE_EXIT	
1	DECIMAL	6	XSF_XDSB_DATA_OCE_DEF	
1	DECIMAL	7	XSF_XDSB_DATA_OCE_MAX	
1	DECIMAL	8	XSF_XDSB_DATA_OCE_VOLCAT	
1	DECIMAL	9	XSF_XDSB_DATA_LCS	
1	DECIMAL	10	XSF_XDSB_DATA_LCS_DEF	
1	DECIMAL	11	XSF_XDSB_DATA_TVEXTPURGE	

Table 21. Constants for XSF_SFI (continued)

Len	Type	Value	Name	Description
1	DECIMAL	12	XSF_XDSB_DATA_CNVT	
1	DECIMAL	13	XSF_XDSB_DATA_EXPORT	
1	DECIMAL	14	XSF_XDSB_DATA_LASTREF	
1	DECIMAL	15	XSF_XDSB_DATA_OCE_MC	
1	DECIMAL	16	XSF_XDSB_DATA_CATRETPD	
1	DECIMAL	17	XSF_XDSB_DATA_CATLG_DAYS	
1	DECIMAL	18	XSF_XDSB_DATA_DEFTABLE	
8	HEX	000C8C700000000A	XSF_SFI_XTM	Last inventory management expiration time
3	HEX	8C7000	XSF_SFI_XTM_ID	
2	HEX	000C	XSF_XTM_LENGTH	
1	HEX	0A	XSF_XTM_DTYPE	
8	HEX	00098C7800000003	XSF_SFI_X100	EDGUX100 exit status
3	HEX	8C7800	XSF_SFI_X100_ID	
2	HEX	0009	XSF_X100_LENGTH	
1	HEX	03	XSF_X100_DTYPE	
8	HEX	00098C7801000003	XSF_SFI_X200	EDGUX200 exit status
3	HEX	8C7801	XSF_SFI_X200_ID	
2	HEX	0009	XSF_X200_LENGTH	
1	HEX	03	XSF_X200_DTYPE	
8	HEX	00098C7802000003	XSF_SFI_X300	EDGUX300 exit status
3	HEX	8C7802	XSF_SFI_X300_ID	
2	HEX	0009	XSF_X300_LENGTH	
1	HEX	03	XSF_X300_DTYPE	
8	HEX	00108C8000000007	XSF_SFI_2JBN	Second VRS job name mask
3	HEX	8C8000	XSF_SFI_2JBN_ID	
2	HEX	0010	XSF_2JBN_LENGTH	
1	HEX	07	XSF_2JBN_DTYPE	
8	HEX	00108C9000000007	XSF_SFI_2NME	Secondary VRS mask
3	HEX	8C9000	XSF_SFI_2NME_ID	
2	HEX	0010	XSF_2NME_LENGTH	
1	HEX	07	XSF_2NME_DTYPE	
8	HEX	000C8CA000000009	XSF_SFI_2SCD	Secondary VRS subchain start date
3	HEX	8CA000	XSF_SFI_2SCD_ID	
2	HEX	000C	XSF_2SCD_LENGTH	
1	HEX	09	XSF_2SCD_DTYPE	
8	HEX	00108CB000000007	XSF_SFI_2SCN	Secondary VRS subchain name
3	HEX	8CB000	XSF_SFI_2SCN_ID	
2	HEX	0010	XSF_2SCN_LENGTH	
1	HEX	07	XSF_2SCN_DTYPE	

Table 22. Cross Reference for XSF_SFI

Name	Offset	Hex Tag	Level
XSF_OUTBUF	0		1
XSF_OUTBUF_BUFLNG	0		2
XSF_OUTBUF_DATALNG	8		2
XSF_OUTBUF_FIELDS	C		2
XSF_OUTBUF_RQDLNG	4		2
XSF_SFI	0		1
XSF_SFI_COMPDATA	8		2
XSF_SFI_COMPENT	8		3
XSF_SFI_COMPHDR	8		3
XSF_SFI_COMPTYPE	8		4
XSF_SFI_COMPTYPE1	8		1
XSF_SFI_COMPVAL	E		2
XSF_SFI_DATA	8		2
XSF_SFI_DTYPE	7		3
XSF_SFI_DTYP1	A		5
XSF_SFI_DTYP2	D		5
XSF_SFI_FACTOR	B		5
XSF_SFI_FIELD1	9		4
XSF_SFI_FIELD2	C		4
XSF_SFI_HD	0		2
XSF_SFI_ID	2		3
XSF_SFI_IDQUAL	4		4
XSF_SFI_IDVAL	2		4
XSF_SFI_LENGTH	0		3
XSF_SFI_LEN1	9		5
XSF_SFI_LEN2	C		5
XSF_SFI_TYPE	5		3

EDGXSF labeling conventions

This topic includes the labeling conventions used in macro EDGXSF. The conventions are provided to assist you until such time as you are able to obtain macro EDGXSF.

Labeling: Begin and End Resource groups

Resource groups, except for VOL and VRS, are defined using this format:

- XSF_SFI_ID_xxxx and XSF_xxxx_LENGTH
- XSF_SFI_ID_Exxxx and XSF_Exxxx_LENGTH

Here is a sample mapping of the Begin and End ACCESS group:

Len	Type	Value	Name
8	HEX	0008021000000000	XSF_SFI_ACCESS
3	HEX	021000	XSF_SFI_ID_ACCESS
2	HEX	0008	XSF_ACCESS_LENGTH

Len	Type	Value	Name
8	HEX	0008021080000000	XSF_SFI_EACCESS
3	HEX	021080	XSF_SFI_ID_EACCESS
2	HEX	0008	XSF_EACCESS_LENGTH

The VOL and VRS groups are defined using this format:

- XSF_SFI_ID_xxx and XSF_xxxGRP_LENGTH
- XSF_SFI_ID_Exxx and XSF_ExxxGRP_LENGTH

Here us a sample mapping of the Begin and End VOL group:

Len	Type	Value	Name
8	HEX	0008036000000000	XSF_SFI_VOLGRP
3	HEX	036000	XSF_SFI_ID_VOL
2	HEX	0008	XSF_VOLGRP_LENGTH
8	HEX	0008036080000000	XSF_SFI_EVOLGRP
3	HEX	036080	XSF_SFI_ID_EVOL
2	HEX	0008	XSF_EVOLGRP_LENGTH

Labeling: Structured field introducers that introduce data

Structured field introducers introduce data and are defined using this format:

- XSF_SFI_xxxx_ID
- XSF_xxxx_LENGTH
- XSF_xxxx_DTYPE

Here is a sample mapping of the ATM SFI:

Len	Type	Value	Name	Description
8	HEX	000C80600000000A	XSF_SFI_ATM	Assigned time
3	HEX	806000	XSF_SFI_ATM_ID	
2	HEX	000C	XSF_ATM_LENGTH	
1	HEX	0A	XSF_ATM_DTYPE	

Labeling: Flags

Output data for some structured field introducers are defined as bit flags using this format: XSF_xxxx_FLAG_name.

Here is a sample mapping of the ACT SFI:

Len	Type	Value	Name	Description
8	HEX	0009802000000002	XSF_SFI_ACT	Actions on release
3	HEX	802000	XSF_SFI_ACT_ID	
2	HEX	0009	XSF_ACT_LENGTH	
1	HEX	02	XSF_ACT_DTYPE	
1	HEX	80	XSF_ACT_FLAG_SCRATCH	

Len	Type	Value	Name	Description
1	HEX	40	XSF_ACT_FLAG_REPLACE	
1	HEX	20	XSF_ACT_FLAG_INIT	
1	HEX	10	XSF_ACT_FLAG_ERASE	
1	HEX	08	XSF_ACT_FLAG_RETURN	
1	HEX	04	XSF_ACT_FLAG_NOTIFY	

Labeling: Bin(8) data

Output data for some structured field introducers are defined as one-byte binary numbers using this format: XSF_xxxx_DATA_name.

Here is a sample mapping of the LOCT SFI:

Len	Type	Value	Name	Description
8	HEX	000984E000000003	XSF_SFI_LOCT	Location type
3	HEX	84E000	XSF_SFI_LOCT_ID	
2	HEX	0009	XSF_LOCT_LENGTH	
1	HEX	03	XSF_LOCT_DTYPE	
1	NUMB HEX	00	XSF_LOCT_DATA_SHELF	
1	NUMB HEX	01	XSF_LOCT_DATA_STORE_BUILTIN_BINS	
1	NUMB HEX	02	XSF_LOCT_DATA_MANUAL	
1	NUMB HEX	03	XSF_LOCT_DATA_AUTO	
1	NUMB HEX	04	XSF_LOCT_DATA_STORE_BINS	
1	NUMB HEX	05	XSF_LOCT_DATA_STORE_NOBINS	
1	NUMB HEX	06	XSF_LOCT_DATA_INCTNR	

Unlabeled data

These output data types are unlabeled:

- Fixed-length and variable-length character data
- Two-byte binary values
- Four-byte binary values
- Dates
- Times

Appendix D. Hexadecimal example of an output buffer

This topic provides an example and discussion of a hexadecimal representation of the contents of an output buffer for a SEARCHDATASET subcommand request. You can modify this example for use in your installation.

Hexadecimal representation of an output buffer

Figure 36 on page 153 is a hexadecimal representation of the contents in an output buffer that might be produced for the SEARCHDATASET VOLUME(VOL001) subcommand shown in "Requesting standard output" on page 39. This format is used:

- Relative buffer address shown as 2-byte values.
- Buffer contents are shown in groups of 8-bytes.

```
0000 0000100000000000 0000008400080260 00000000001A82A0 00000007D9D4D4E4
0020 E2C5D94BC6C9C5D3 C44BE3C5E2E3000E 8BC000000001E5D6 D3F0F0F1000F8700
0040 00000007D9D4D4E4 E2C5D9000C8A9E00 000005FFFF9D9000 0C81300000000920
0060 05320F000C81A000 00000A0658226F00 0C833000000000500 0000010008026080
0080 0000000000000000 0000000000000000 0000000000000000 0000000000000000

0FFC 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0FFE 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

Figure 36. Hexadecimal representation of the contents of an output buffer

Description of the contents of an output buffer

The first line of the output buffer shown in Figure 36 on page 153 shows:

```
0000 0000100000000000 0000007100080260 00000000001B82A0 00000007D6E6D5C5
```

- Three 4-byte length fields:
  - 00001000

    This is the length you specified for the output buffer.
  - 00000000

    This means that the output buffer is large enough. When the buffer length is too small, DFSMSrmm sets this field with the size of the buffer needed. DFSMSrmm also returns return code 108 and reason code 10.
  - 00000084

    This is the total size of the data in the output buffer, including the length of this field. You can use this data length to determine when there is no more data to process.
- Eight structured fields:
  - 0008026000000000

    This is the Begin DATASET group SFI, which begins at offset x'000C' into the output buffer. Use this SFI to confirm that you are processing a DATASET SFI. When you do not want to process a group of structured fields, scan to the end of the group by looking for the corresponding End SFI, such as, the End DATASET group SFI in this example.

The first and second lines of the output buffer shown in Figure 36 on page 153 show:

```
0000 0000100000000000 0000007100080260 00000000001B82A0 00000007D6E6D5C5
0020 D9D6D5C54BC6C9C5 D3C44BE3C5E2E300 0E8BC000000001E5 D6D3F0F0F1001087
```

- Data Set Name structured field
  - 001B82A000000007 D6E6D5C5D9D6D5C54BC6C9C5D3C44BE3C5E2E3

    This is the Data Set Name structured field, which begins at offset x'0014' into the output buffer. The structured field consists of the 8-byte DSN SFI and, in this example, the 19-byte data set name (OWNERONE.FIELD.TEST). The length of the structured field is 27 bytes (8 plus 19) as shown by the x'001B' value at the beginning of the field.
- Volume Serial structured field
  - 000E8BC000000001 E5D6D3F0F0F1

    This is the Volume Serial structured field, which begins at offset x'002F' into the output buffer. The structured field consists of the 8-byte VOL SFI and the 6-byte volume serial (VOL001).

The second and third lines of the output buffer shown in Figure 36 on page 153 show:

```
0020 D9D6D5C54BC6C9C5 D3C44BE3C5E2E300 0E8BC000000001E5 D6D3F0F0F1001087
0040 0000000007D6E6D5 C5D9D6D5C5000C81 3000000009199711 7C000C81A0000000
```

- Owner structured field
  - 0010870000000007 D6E6D5C5D9D6D5C5

    This is the Owner structured field, which begins at offset x'003D' into the output buffer. The structured field consists of the 8-byte OWN SFI and the 8-byte owner (OWNERONE).
- Create Date structured field
  - 000C813000000009 1997117C

    This is the Create Date structured field, which begins at offset x'004D' into the output buffer. The structured field consists of the 8-byte CDTJ SFI and the 4-byte packed-decimal date (x'1997117C').

The third and fourth lines of the output buffer shown in Figure 36 on page 153 show:

```
0040 0000000007D6E6D5 C5D9D6D5C5000C81 3000000009199711 7C000C81A0000000
0060 0A0815270C000C83 3000000005000000 0100080260800000 0000000000000000
```

- Create Time structured field
  - 000C81A00000000A 0815270C

    This is the Create Time structured field, which begins at offset x'0059' into the output buffer. The structured field consists of the 8-byte CTM SFI and the 4-byte packed-decimal time (x'0815270C').
- Physical File Sequence structured field
  - 000C833000000005 00000001

    This is the Physical File Sequence structured field, which begins at offset x'0065' into the output buffer. The structured field consists of the 8-byte FILE SFI and the 4-byte binary sequence number (x'00000001').
- End DATASET group SFI
  - 0008026080000000

    This is the End DATASET group SFI, which begins at offset x'0071' into the output buffer.

Processing the contents of an output buffer

To process the contents of an output buffer, consider using these guidelines:

1. Base the XSF_OUTBUF definition in macro EDGXSF as shown in Figure 37 on page 155 on the address of the output buffer you are interested in.

```
XSF_OUTBUF           DSECT    Output Buffer
XSF_OUTBUF_BUFLNG    DS       1FL4    Buffer Length
XSF_OUTBUF_RQDLNG    DS       1FL4    Required Buffer Length
XSF_OUTBUF_DATALNG   DS       1FL4    Length of Output Data
XSF_OUTBUF_FIELDS    DS       0C      Start of Structured Fields
```

Figure 37. Output buffer definition

2. Base the XSF_SFI definition in macro EDGXSF as shown in Figure 38 on page 155 on the address of XSF_OUTBUF_FIELDS.

```
XSF_SFI         DSECT          Structured Field Introducers
XSF_SFI_LENGTH  DS   1FL2      Length
XSF_SFI_ID      DS   1CL0003   ID (identifier)
                ORG  XSF_SFI_ID
XSF_SFI_IDVAL   DS   1CL0002   ID (Identifier Value)
XSF_SFI_IDQUAL  DS   1CL0001   ID (Identifier Qualifier)
XSF_SFI_TYPE    DS   1FL1      Type
                DS   1CL0001   Reserved
XSF_SFI_DTYPE   DS   1FL1      Data type
XSF_SFI_LEN     EQU  *-XSF_SFI
XSF_SFI_DATA    DS   0C        Start of Data
```

Note: XSF_SFI_DATA can contain compound data with an internal structure of:

```
XSF_SFI_CompType
XSF_SFI_LEN1
XSF_SFI_DTYPE1
XSF_SFI_Factor
XSF_SFI_LEN2
XSF_SFI_DTYPE2
XSF_SFI_Value
```

Figure 38. SFI definition

3. Find the type of structured field you are processing by using the two-byte structured field identifier at XSF_SFI_IDVAL. The values of XSF_SFI_IDQUAL for ADL, address line SFI, and UID, User ID SFI, described in Appendix A, “Structured field introducers (SFIs),” on page 63 are not constant values.
4. Move to the next structured field by adding the length at XSF_SFI_LENGTH to the XSF_SFI pointer.
5. Verify that you have reached the end of the valid data in the output buffer by using the length of the output data at XSF_OUTBUF_DATALNG.
6. Determine the type of data you are processing, by using the value in XSF_SFI_DTYPE.
7. Obtain the length of the data that starts at XSF_SFI_DATA, by subtracting XSF_SFI_LEN from the structured field length at XSF_SFI_LENGTH. in the output buffer.
8. Move to the end of the SFI by adjusting the pointer. In this example, when your pointer is at offset x'00000071' into the output buffer, there are two indicators that you are done with the contents of the buffer:
   - You are looking at the End DATASET group SFI.

     Note: This is true only if you did not specify MULTI=YES in your call to the API. If you use MULTI=YES, your output buffer may contain more than one resource group.
   - Adjusting the XSF_SFI pointer by the length of this SFI (8 bytes) points you past the last byte of data in the buffer.
9. Repeat these steps to process each structured field.

In the examples shown in Figure 37 on page 155 and Figure 38 on page 155:

- Adding the length of the data (x'00000071') at XSF_OUTBUF_DATALNG to the address of XSF_OUTBUF_DATALNG results in the address just beyond the last byte of data in the output buffer. You might find this a useful double-check to ensure that you are looking at valid data.

- Your XSF_SFI pointer is at the first structured field in the output buffer (offset 000C in the buffer), and the SFI identifier value at XSF_SFI_IDVAL (0260) tells you that the SFI is a Begin DATASET group. To move to the next structured field, add XSF_SFI_LENGTH (0008) to your pointer.
- Your XSF_SFI pointer is now at the second structured field in the output buffer (offset 0014 in the buffer); XSF_SFI_IDVAL (82A0) identifies the SFI as DSN (Data Set Name); and XSF_SFI_LENGTH (001B) minus XSF_SFI_LEN (8) gives you a length of 19 bytes for the data set name. The type of data is variable-length character because the data type at XSF_SFI_DTYPE equals XSF_SFI_DTYPE_CHAR_VAR.

  One method to process structured field introducers is to use an SFI lookup table containing ID values and addresses of corresponding processing routines. Another method is to use the XSF_SFI_DTYPE: Call an appropriate data-type routine with the address of the SFI or SFI data and the address of an output area as inputs.

  After you finish processing this structured field, update the XSF_SFI pointer to the next structured field.

Appendix E. Accessibility

Accessible publications for this product are offered through IBM Documentation (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) 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

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)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

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

Trademarks

DFSMSrmm
IBM
IBMLink
RACF
z/OS
z/VM®

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

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

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.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Product Number: 5650-ZOS

SC23-6872-50