

SAMS:Compress[®] for IMS

Comprehensive Data Compression System

User's Guide

Version 5.2

February 1994

© Copyright 1991, 1994
Sterling Software, Inc.
All Rights Reserved

Sterling Software, Inc.
Storage Management Division
202 East Airport Drive, Suite 280
San Bernardino, California 92408-3429

SHRIM52A-M

Proprietary/Trade Secret Notice

This documentation, the software it describes, and the information and know-how they contain constitute the proprietary, confidential and valuable trade secret information of Sterling Software, Inc., and/or of the respective manufacturer or author, and may not be disclosed to others without the prior written permission of Sterling Software, Inc. This documentation and the software which it describes have been provided pursuant to a license agreement which contains prohibitions against and/or restrictions on their copying, modification and use. Reproduction, in whole or in part, if and when permitted, shall bear this notice and the Sterling Software, Inc. copyright notice. As and when provided to any governmental entity, government contractor, or subcontractor subject to the Federal Acquisition Regulations, this documentation is provided with RESTRICTED RIGHTS.

Warranty Disclaimer, Right to Revise

This documentation and the software which it describes are licensed either "AS IS" or with a limited warranty, as set forth in the applicable license agreement. Other than any limited warranties provided, NO OTHER WARRANTY IS EXPRESSED AND NONE SHALL BE IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE OR FOR A PARTICULAR PURPOSE.

Sterling Software, Inc. reserves the right to revise this publication from time to time and to make changes in the content hereof without the obligation to notify any person or entity of such revisions or changes.

Trademarks

CICS, COBOL, DB2, IDCAMS, IMS, ISAM, ISPF, JES, MVS, OSAM, PL/I, and VSAM are either trademarks or registered trademarks of the International Business Machines Corporation.

SAMS:Compress[®] and Shrink[®] are registered trademarks of Sterling Software, Inc.

Abend-Aid[®] is a registered trademark of Compuware.

Technical Support

For Users in the United States

For technical support please contact:

*Sterling Software, Inc.
Storage Management Division
11050 White Rock Road, Suite 100
Rancho Cordova, CA 95670-6095
(800) 889-0226*

For Users Outside the United States

For technical support please contact your local Sterling Software customer support representative.

Table of Contents

Proprietary/Trade Secret Notice	ii
Warranty Disclaimer, Right to Revise	ii
Trademarks	ii
Technical Support	iii
For Users in the United States	iii
For Users Outside the United States	iii

Chapter 1. Introduction **1**

About This User's Guide	1
Table of Contents	1

Chapter 2. Installation **5**

Overview	5
SAMS:Compress for IMS Installation Checklist	5
All SAMS:Compress Systems	5
The Interactive Dialog	6
SAMS:Compress for IMS and SAMS:Compress for IMS Express	6
SAMS:Compress for IMS Express Only	6
Support for IMS Fastpath Unload/Reload Utilities (Optional)	7
The Callable Subroutines for use with CICS (Optional)	7
Transferring the JCL from the Tape	8
Loading the JCL Library	8
Loading the SAMS:Compress Load Library	10
Installing the Interactive Dialog	10
System Upgrading Considerations for SAMS:Compress 5.2	11
Loading the Interactive Dialog Libraries	12
Allocating GENLIB and FDTLIB Libraries	13
Initializing the Interactive Dialog Control File	13
User Modifications to the Interactive Dialog Libraries	17
Concatenating the Interactive Dialog Libraries	21
Activating the Interactive Dialog	22
Authorizing a User for the Interactive Dialog	24
Updating the User Data Information in the Control File	24
Deleting an Authorized User From the Control File	24
Listing the Interactive Dialog Authorized Users	24
Installing SAMS:Compress for IMS	25
Loading the SAMS:Compress for IMS Library	25
Setting the Active File Descriptor Table Parameter	25
Generating the SAMS:Compress Program Modules	26
System Upgrading Considerations	29
Online Environment	29
Batch Processing	30

Installing SAMS:Compress for IMS Express	30
Loading the Express Library	30
Generating the Express Program Modules	31
Online Environments	33
Batch Processing	33
Installing Support for IMS Fastpath Unload/Reload Utilities (Optional)	34
Installing the Callable Subroutines for Use With CICS (Optional)	35
Loading the Source Modules	35
Specify the FDT Names and a Fullword in the CWA	36
Generating the SAMS:Compress Program Modules	37
Assembling and Linking the Program Modules	39
CICS and Application Program Requirements	39

Chapter 3. Implementation Considerations 41

Overview	41
Features	41
Guidelines for Choosing Data Bases to Compress	42
Implementation for an IMS Data Base	42
Implementation Steps Outline	42
Identify an IMS Data Base to SAMS:Compress	43
Compress an IMS Data Base With SAMS:Compress	49
Obtaining the Best Compression for a Data Base	50
Considerations in Specifying Compression Controls	52
Entering or Updating the RDL for a Data Base Segment	53
Controlling the Size of a Compressed Segment	55
Deleting an IMS Data Base From the Control File	56
Changing a Data Base Controlled by SAMS:Compress	56

Chapter 4. Implementing Express 59

Overview	59
Express Features	59
Implementing Express for an IMS Data Base	60
Implementation Steps Summary	60
Identify an IMS Data Base to Express	60
Compress an IMS Data Base With Express	63
Deleting an IMS Data Base From the Control File	64
Changing a Data Base Under the Control of Express	64

Chapter 5. Interactive Dialog Reference 65

Recreating a Lost or Damaged FDT	66
SAMS:Compress Interactive Dialog Panel - SFEP\$000	67
Unknown User ID Panel - SFEP\$UUI	68
Report Printing Option Panel - SFEPHCPY	69
Control File I/O Error Panel - SFEPCFIO	70
Data Set Allocation Error Panel - SFEPALER	72
Primary Services Menu - SFEPMENU	74

FDT Support Services Menu - SFEP8MNU	75
FDT Regeneration Confirmation Panel - SFEP8FGC	76
FDT Deletion Confirmation Panel - SFEP8DDC	77
Control Maintenance Menu - SFEP9MNU	78
Maintain Installation Control Parameters Panel - SFEP9CMP	79
User Maintenance Menu - SFEP9UID	81
User Data Maintenance Panel - SFEP9USR	82
User Deletion Confirmation Panel - SFEP9UDC	83
Define Default Data Sets Panel - SFEP9DSN	84
IMS Support Services Menu - SFEP1MNU	85
IMS Data Base Analysis Panel - SFEP1ANA	86
IMS DBD Analysis Duplicate DDNAME Confirmation Panel - SFEPDUPD	87
IMS Database Stored Data Report Generation Panel - SFEP1RPT	88
IMS User Parameter Maintenance Menu - SFEP1UPM	89
IMS Database Title Maintenance Panel - SFEP1UDT	90
IMS Data Sample Extract Parameter Maintenance Panel - SFEP1UEP	91
RDL Maintenance Selection List Panel - SFEP1URL	92
IMS User RDL Parameter Maintenance Panel - SFEP1URP	93
IMS User FDT Parameter Maintenance Panel - SFEP1UFP	94
IMS User Data Set Name Maintenance Panel - SFEP1UDS	95
IMS Database Segment Count Maintenance Panel - SFEP1USC	96
IMS JCL Generation Menu - SFEP1JCL	97
IMS JCL Generation Menu (Full Function) - SFEP1JFF	98
IMS JCL Generation Menu (FASTPATH) - SFEP1JFP	99
IMS Data Extraction JCL Generation (Full Function) Panel - SFEP1JD1	100
IMS Data Extraction JCL Generation (BMP) Panel - SFEP1JDA	101
IMS Data Extraction JCL Generation Panel, Part 2 - SFEP1JEF	102
IMS Data Base Implementation Menu- SFEP1IMN	104
IMS Segment Selection Panel - SFEP1FDT	105
IMS DBD Generation Panel - SFEP1DBD	106
IMS Delete Data Base From the Control File Panel - SFEP1DEL	107
IMS DBD Data Deletion Confirmation Panel - SFEP1DDC	108

Chapter 6. SAMS:Compress/2 **109**

Overview	109
Features	109
Modes of Use	111
Using the File Utility Programs	112
File Prepass Utility	113
File Descriptor Table	113
Execution Example–File Prepass Utility	114
Prepass Statistics	115
FDTLOADR Utility	117
Advantages of Use of FDTs in Load Module Format	117
Execution Example–FDTLOADR Utility	118
File Compression Utility	118
Considerations Before Compressing	119
Execution Example–File Compression Utility	120

File Compression Utility With FDT in Load Module Format	120
Compression Statistics	121
Combining File Prepass and File Compression in a Single Job Step	123
File Expansion Utility	124
Execution Example–File Expansion Utility	124
File Expansion Utility With FDT in Load Module Format	125
Default JCL for the Compression and Expansion Utilities	125
The DSORG Parameter	126
The RECFM Parameter	126
The LRECL Parameter	126
The BLKSIZE Parameter	126
The KEYLEN Parameter	126
The RKP Parameter	127
The AMP Parameter	127
Express File Expansion Utility	127
Execution Example–Express File Expansion Utility	127
Using SAMS:Compress/2 Subroutines	128
Accessing the FDT	129
JCL Implications for Existing Application Programs	130
Calling the Subroutines	130
User Considerations	131
General Sequence for Subroutine CALLs	131
CALL to Subroutine SHRINK	133
CALL to Subroutine EXPAND	134
CALL to Subroutine PUFFUP	135
CALL to Subroutine PUFFDOWN	136
CALL to Subroutine CLOSE	138
Incorporating Subroutine Calls in Existing Application Programs	139
Defining Compressed Records in COBOL Application Programs	140
Linking Subroutines With Application Programs	141
Using SAMS:Compress/2 Under CICS	142
Install the Callable SHRINK Subroutines for CICS	142
Callable SHRINK Subroutines for CICS Install Checklist	143
Load the Source Modules	143
Specify the FDT Names and a Fullword in the CWA	143
Assemble and Link the Program Modules	144
Satisfy CICS and Application Program Requirements	145
SAMS:Compress/2 Subroutines Under CICS	145
SCB initialization	146
Calling Sequences for SAMS:Compress/2	146
VARISAM–Variable-length ISAM Subroutines	149
Converting COBOL Programs to Use VARISAM Subroutines	149
VARISAM Subroutine Descriptions and Coding Instructions	152
Link-Editing VARISAM Subroutines With User Application Programs	157

Chapter 7. Record Definition Language 161

Overview	161
Performance Considerations When Using RDL	162

How the RDL Operates	162
RDL Terminology	163
RDL Syntax	164
RDL Field Type Descriptions	166
Field Types C1, C2, and C3—Character Data	166
Field Type GA—Garbage Data (Permanently Unused Fields)	167
Field Type L—Insert Tally of Actual Length	167
Field Types MA and MB—Pattern Matching	168
Field Type N—Fields Exempted From Compression	169
Field Type PD—Packed Decimal Data	170
Field Types S and X—Set of Expected Values	171
Field Type UN—Undefined Fields	172
Field Types V, VP, and VZ—Calculating a Value for the Variable Symbol	173
Field Types ZL and ZR—Zoned Decimal Data	174
RDL Repetition Groups	175
RDL Condition Groups	176
RDL Position Function	179
General Restrictions on RDL Use	182
Guide to Correct RDL Specifications	183
RDL Defaults	185

Chapter 8. Implementing SAMS:Compress Without the Interactive Dialog 187

Implementation Overview	187
Obtaining Optimal Compression	189
User Constraints	189
Data Base Integrity	189
Defining the Data Base to SAMS:Compress for IMS	190
Procedural Overview	190
Describing Segments With RDL	190
Defining Fixed-Length Segments	192
Defining Variable-Length Segments	193
Executing the IMSPASS Utility	193
IMSPASS Parameters	195
Sharing an FDT	196
Using the OUTDB Data Set	197
Executing the FDTLOADR Utility	198
Performing Data Base Compression	199
Procedural Overview	199
Data Base Reorganization/Unload	200
DBD Generation	200
Coding the SEGM Macro	201
Coding the SEGCC Macro	201
DBD Control Statement Assembly	203
Data Base Reload	203

Chapter 9. Implementing Express Without the Interactive Dialog	205
Implementation Overview	205
Method I: Editing Existing DBD Source Macros	205
Method II: Using a Data Dictionary to Generate DBD Source Macros	206
Chapter 10. Reports	207
Reports Common to All SAMS:Compress Products	208
FDT Index Listing	208
Authorized User List	209
IMS Reports	210
Index List	210
Stored Data Report	211
Extracted Data Sample Report	225
Byte Distribution Analysis Report	228
Test Compression Report	231
Chapter 11. Messages and Abend Codes	255
Overview	255
Abend Codes	255
Messages with Message IDs	287
Common Error Messages	287
RDL Error Messages	296
SAMS:Compress for IMS Error Messages	299
Express Error Messages	306
Interactive Dialog Error Messages and Abend Codes	307
Messages Without Message IDs	320
VARISAM Error Messages	320
SEGCC Error Messages (DBDGEN)	323
Appendix A. Customer Support Services	325
What to do When You Call In	326
Tell us about your organization and yourself	326
Your environment	326
The problem	326
Collect the documents that describe the problem	326
What Happens to Reported Problems?	326
Glossary	327
Index	331

List of Figures

Figure 2-1. JCL to Load the Installation JCL Data Set	9
Figure 2-2. JCL to Load the Load Modules	10
Figure 2-3. JCL to Load the Dialog Libraries	12
Figure 2-4. Allocating GENLIB and FDTLIB	13
Figure 2-5. Control Statements for VSAM Cluster Def.	14
Figure 2-6. JCL to Initialize Dialog Control File	15
Figure 2-7. Panel SFEP\$UUI to Define TSO Users	17
Figure 2-8. Extract JCL for Better BDA and Comp. Tests	18
Figure 2-9. Sort Control Statements (SFEEEXSCD)	18
Figure 2-10. VSAM Buffer Pool Control Statements	18
Figure 2-11. CLIST to Initialize the Dialog (SHRIMS)	20
Figure 2-12. Allocating ISPF Data Sets, TSO LOGON proc	21
Figure 2-13. Allocating ISPF Data Sets, CLIST	21
Figure 2-14. JCL to Load the SAMS:Compress Source Lib.	25
Figure 2-15. JCL to Set FDT Value in SHRINKCB	26
Figure 2-16. JCL to Assemble and Link Modules	27
Figure 2-17. JCL to Load the Express Data Set	30
Figure 2-18. JCL to Assemble and Link IMEXPRES	31
Figure 2-19. JCL to Load the Load Modules	34
Figure 2-20. JCL to Load the Callable Subroutines	35
Figure 2-21. JCL to Set the CWA Fullword Offset and FDT	36
Figure 2-22. JCL to Assemble and Link Source Modules	37
Figure 6-1. SAMS:Compress/2 Operation	111
Figure 6-2. SHRINK File Prepass Utility Job Step	115
Figure 6-3. Field Type C Table	116
Figure 6-4. FDTLOADR Utility Job Step	118
Figure 6-5. File Compression Utility—FDT in Sequential Data Set Format	120
Figure 6-6. File Compression Utility—FDT in Load Module Format	121
Figure 6-7. Compression Statistics Produced by the File Compression Utility	122
Figure 6-8. Combined Execution of Prepass and Compression Utilities	123
Figure 6-9. File Expansion Utility—FDT in Sequential Data Set Format	124
Figure 6-10. File Expansion Utility—FDT in Load Module Format Pass the data address	125
Figure 6-11. Express File Expansion Utility	128
Figure 6-12. General Flow of Subroutines	132
Figure 6-13. The PUFFUP Process	135
Figure 6-14. The PUFFDOWN Process	136
Figure 6-15. Application Link-Edit with Subroutines	142
Figure 6-16. COBOL—Processing of Fixed-Length Uncompressed ISAM Files	150
Figure 6-17. COBOL—Converted to Use VARISAM to Process Compressed Variable-Length ISAM Files	151
Figure 6-18. VARISAM Link-Edit to Application—For Version Before 3.0	158
Figure 6-19. VARISAM Link-Edit with User Application	159
Figure 7-1. Invoicing File Record Set and Field Descriptions	177
Figure 7-2. Name and Address File Record Set and Field Descriptions	180
Figure 8-1. SAMS:Compress for IMS Operation	188
Figure 8-2. Defining the Data Base to SAMS:Compress	191

Figure 8-3. RDL for Fixed-Length Segments	192
Figure 8-4. RDL for Variable-Length Segments	193
Figure 8-5. IMSPASS Execution JCL	197
Figure 8-6. File Prepass Utility JCL for RDL Revision	198
Figure 8-7. FDTLOADR Utility JCL	199
Figure 8-8. IMSHRINK Operation	200
Figure 8-9. Sample DBD Control Statements	202
Figure 10-1. FDT Index Listing	208
Figure 10-2. Authorized User List	209
Figure 10-3. Index List	210
Figure 10-4. Page Heading and Database Information Section	211
Figure 10-5. Root Segment	213
Figure 10-6. Dependent Segment	217
Figure 10-7. Data Set Information Section	221
Figure 10-8. JCL Generation Information Section	223
Figure 10-9. Heading Section	225
Figure 10-10. Data Section	226
Figure 10-11. Heading Section	228
Figure 10-12. Detail Section	229
Figure 10-13. Heading for IMS Full Function Database—Single DSG	232
Figure 10-14. Heading for IMS Full Function Database—Multiple DSGs	233
Figure 10-15. Compression Test Information Section—IMS Database	234
Figure 10-16. Multiple DSGs Database—OSAM	236
Figure 10-17. Single DSG HIDAM Database—VSAM	239
Figure 10-18. HISAM Database—VSAM	241
Figure 10-19. Multiple DSG HDAM Database—OSAM	243
Figure 10-20. Single DSG HIDAM Database—VSAM	245
Figure 10-21. HISAM Database—VSAM	246
Figure 10-22. DASD Usage and Savings Sub-section	248
Figure 10-23. CPU Performance Sub-Section	250
Figure 10-24. Installation Computation Factors Sub-Section	252

List of Tables

Table 2-1. The Contents of the Installation Tape	8
Table 6-1. Calculation of LRECL for Compressed Data Sets	119
Table 6-2. Parameters for Subroutine SHRINK	133
Table 6-3. Parameters for Subroutine EXPAND	134
Table 6-4. Parameters for Subroutine PUFFUP	136
Table 6-5. Parameters for Subroutine PUFFDOWN	138
Table 6-6. Parameters for Subroutine SHRINK	148
Table 6-7. Parameters for Subroutine EXPAND	148
Table 7-1. SAMS:Compress Field Types	165
Table 7-2. Field Length Descriptors	166
Table 7-3. FDT Space Requirements of RDL Specifications	182
Table 7-4. RDL Default Specifications for Sequential Files	185
Table 7-5. RDL Default Specifications for ISAM Files	185
Table 7-6. RDL Default Specifications for VSAM Files	186
Table 8-1. IMSPASS Data Sets	194
Table 8-2. IMSPASS Parameter Specifications	195

Chapter 1. Introduction

The SAMS:Compress Data Compression and Encryption System is a proprietary computer software product of Sterling Software, Inc. The SAMS:Compress software operates under control of the IBM OS or OS/VS Operating System.

The SAMS:Compress product interfaces contain unique features, but provide comparable performance capabilities. The interfaces that use Huffman compression routines can consistently realize a compression rate of 50 to 60 percent. Often the compression yield is as high as 70 to 80 percent with Huffman. The Express compression products offer lower compression rates while limiting CPU overhead. All members of the SAMS:Compress product line use a check byte comparison to ensure complete data integrity. And, all products are developed, maintained and supported by Sterling Software, Inc.

About This User's Guide

The SAMS:Compress User's Guide contains documentation for SAMS:Compress, SAMS:Compress Express, and the callable SAMS:Compress subroutines. It is organized as follows:

Table of Contents

Introduction

This chapter provides an overview and history of the SAMS:Compress family of products, a description of the documentation, and instructions for contacting the SAMS:Compress Customer Services department.

Installation

This chapter provides all the information you need to install SAMS:Compress. It is directed to the systems programmer who is responsible for installing new products.

Implementation Considerations

This chapter provides instructions for implementing SAMS:Compress. It also contains the necessary information for implementing SAMS:Compress Express. It is directed to the IMS data base administrator.

Implementing Express

This chapter provides instructions for implementing SAMS:Compress Express. It is directed to the IMS data base administrator.

Interactive Dialog Reference

This chapter contains illustrations of all Interactive Dialog screens with instructions for their use.

SAMS:Compress/2

This chapter provides instructions for using SAMS:Compress/2, the basic SAMS:Compress product. As a group of data set utility programs, SAMS:Compress/2 can be used to compress and expand an entire data set. As subroutines, SAMS:Compress/2 modules can be called from application programs written in Assembler Language or high-level languages. This chapter is directed to application programmers who use SAMS:Compress/2 to compress their data sets.

This chapter also includes instructions for using SAMS:Compress/2 for data sets under the control of CICS applications programs. It also provides a set of subroutines, collectively termed VARISAM, to replace PROCEDURE DIVISION input/output statements which operate upon compressed ISAM data sets, to enable COBOL users of ISAM files to process compressed ISAM data sets.

Record Definition Language

This chapter describes SAMS:Compress's Record Definition Language, which provides the user with a formal means of describing characteristics of the data comprising a file that is compressed (and expanded) by SAMS:Compress. Users desiring to achieve maximum compression may customize RDL for files or data bases under the control of any of the SAMS:Compress products.

Implementing SAMS:Compress Without the Interactive Dialog

This chapter provides documentation for users who are not able to use the facilities of ISPF for the SAMS:Compress Interactive Dialog to implement the SAMS:Compress interface.

Implementing Express Without the Interactive Dialog

This chapter provides documentation for SAMS:Compress Express users who do not wish to use the SAMS:Compress Interactive Dialog.

Reports

This chapter provides samples of all SAMS:Compress reports, with explanations of the fields that appear on them.

Messages and Abend Codes

This chapter contains a complete listing of all messages produced by the SAMS:Compress products, and provides a more complete description of why the message was issued and what actions should be taken by the user.

Appendix A: Customer Support Services

This appendix describes how to contact Customer Support and the information you need when you call

Glossary

Index

Chapter 2. Installation

Overview

This chapter describes how to install the SAMS:Compress system. Instructions are provided for SAMS:Compress, SAMS:Compress Express, SAMS:Compress/2, and the SAMS:Compress Interactive Dialog.

You will find a checklist summarizing the steps you will follow depending upon which SAMS:Compress interface you are installing. After the checklist are detailed instructions and comments for each step, including an illustration of the JCL you will need to execute for each step, and an explanation of the steps which are optional or which can be implemented in different ways.

First, review the installation steps in the checklist and the expanded discussion, marking those steps which are necessary for your configuration.

SAMS:Compress Express is a subset of the SAMS:Compress software system. Therefore, if you are installing the entire system on your mainframe, SAMS:Compress Express is included in the installation steps found in the SAMS:Compress chapter. If you are installing the SAMS:Compress Express software system, you need to skip the SAMS:Compress chapter and follow the installation instructions found in the SAMS:Compress Express chapter.

SAMS:Compress for IMS Installation Checklist

All SAMS:Compress Systems

<u>Complete</u>	<u>Activity</u>
<input type="checkbox"/>	Load the PDS containing the SAMS:Compress JCL from tape to disk. Use the example JCL illustrated in Figure 2-1.
<input type="checkbox"/>	Load the SAMS:Compress Load Library data set from tape to disk. Use the JCL in INSTALL1.

The Interactive Dialog

If you are not running ISPF Version 2.0 or higher in your environment, skip this section. The Dialog does not support ISPF below version 2.0.

Complete	Activity
<input type="checkbox"/>	Considerations before performing system upgrades.
<input type="checkbox"/>	Load the Interactive Dialog Library Data Sets from tape to disk. Use the JCL in INSTAL20.
<input type="checkbox"/>	Allocate PDS libraries for GENLIB and FDTLIB. Use the JCL in INSTAL21.
<input type="checkbox"/>	Initialize the Interactive Dialog Control File and define a primary TSO user to maintain the system. Use the JCL in INSTAL22.
<input type="checkbox"/>	Modify the selected Dialog members.
<input type="checkbox"/>	Activate the Interactive Dialog.
<input type="checkbox"/>	Authorize additional TSO users to use the Interactive Dialog.

SAMS:Compress for IMS and SAMS:Compress for IMS Express

Complete	Activity
<input type="checkbox"/>	Load the SAMS:Compress for IMS source library data set from tape to disk. Use the JCL in INSTALL7.
<input type="checkbox"/>	Set the Active File Descriptor Table (FDT) parameter in SHRINKCB if necessary. Use the JCL in INSTALL8. (Optional).
<input type="checkbox"/>	Considerations before performing system upgrades.
<input type="checkbox"/>	Assemble and link-edit the SAMS:Compress for IMS modules. Use the JCL in INSTALL9 to complete i., ii., and iii.: <ol style="list-style-type: none"> i. Assemble and link-edit the IMSPASS utility module. ii. Assemble and link-edit the IMSHRINK module. iii. Assemble and link-edit the SHRINKCB module.
<input type="checkbox"/>	Load the Express source library data from tape to disk. Use the JCL in INSTAL16.
<input type="checkbox"/>	Assemble and link-edit the Express modules. Use the JCL in INSTAL17.

SAMS:Compress for IMS Express Only

Complete	Activity
<input type="checkbox"/>	Load the Express source library data set from tape to disk. Use the JCL in INSTAL16.
<input type="checkbox"/>	Assemble and link-edit the Express modules. Use the JCL in INSTAL17.

Support for IMS Fastpath Unload/Reload Utilities (Optional)

You may complete the steps in this section if you wish to have compression support in the non-IMS environment Fastpath utility jobs.

Complete	Activity
<input type="checkbox"/>	Load the Fastpath utility support PDS load library from tape to disk. Use the JCL in INSTAL23.
<input type="checkbox"/>	Concatenate this library to the STEPLIB DD statement in any Fastpath Unload/Reload utility jobs for databases which also use SAMS:Compress.

The Callable Subroutines for use with CICS (Optional)

You may complete the steps in this section if you wish to call the SAMS:Compress subroutines from programs running under CICS. This procedure is not required for you to use SAMS:Compress for data sets within a CICS region.

Complete	Activity
<input type="checkbox"/>	Load the SAMS:Compress/2 source library data set from tape to disk. Use the JCL in INSTAL13.
<input type="checkbox"/>	Set the CWASHRK value to point to a reserved fullword in the CWA. Change the member FDTNAMES to list the FDTs to be used. Use the JCL in INSTAL14 to do both.
<input type="checkbox"/>	Assemble and link-edit SAMS:Compress/2 source modules. Use the JCL in INSTAL15.
<input type="checkbox"/>	Modify the CICS tables appropriately. Do both i. and ii.: <ol style="list-style-type: none"> i. Define the SHRNKMOD and SHRKSCBS modules and all FDT modules to be used in the CICS Processing Program Table (PPT). ii. Define SHRNKMOD in the CICS Program Load Table (PLT).
<input type="checkbox"/>	Modify the application programs calling SAMS:Compress/2 CICS subroutines and link-edit them with the SHRKCICS module.

Transferring the JCL from the Tape

The SAMS:Compress installation tape is a standard label 3480 cartridge or 9-track tape written at 6250 BPI, unless otherwise noted on the external tape label. The tape contains the files shown in Table 2-1. The tape you receive contains data for the version of SAMS:Compress you have ordered, while files you don't need are null data sets.

Table 2-1. The Contents of the Installation Tape

File	Content	DCB of Tape Data Set
1	SHRINK.JCL	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
2	SHRINK.LOAD	DSORG=PO,RECFM=U,BLKSIZE=7294
3	SHRINK.SHR2.SOURCE	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=800
4	SHRINK.MVS.LOAD	DSORG=PO,RECFM=U,BLKSIZE=7294
5	SHRINK.IMS.SOURCE	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
6	SHRINK.IDMS.SOURCE	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6400
7	SHRINK.IDMS.LOAD	DSORG=PO,RECFM=U,BLKSIZE=7294
8	SHRINK.ISPF.CLIST	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
9	SHRINK.ISPF.ISPLLIB	DSORG=PO,RECFM=U,BLKSIZE=6144
10	SHRINK.ISPF.ISPPLIB	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
11	SHRINK.ISPF.ISPSLIB	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
12	SHRINK.ISPF.ISPMLIB	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
13	SHRINK.IMEXP.SOURCE	DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=6000
14	SHRINK.IMSUT.LOAD	DSORG=PO,RECFM=U,BLKSIZE=6144

Loading the JCL Library

The first file of the tape contains the JCL procedures used to install the SAMS:Compress system. Transfer this file to a convenient storage medium, such as a disk data set, so you can use it during the installation.

The JCL procedure in Figure 2-1 on the next page may be used to transfer this file to disk. You will need to enter this JCL to perform this procedure. Once this file is on disk, subsequent SAMS:Compress JCL procedures are available as individual members in the partitioned data set SHRINK.JCL.

Note: In this JCL, and in all of the JCL procedures provided, the information shown in lower-case letters should be modified to conform with your installation standards and configuration. Once you modify the JCL for your local standards, save it in case you ever need to reinstall the SAMS:Compress system.

```

/*LOADJCL JOB
/*
/*
/******
/*
/* THIS STEP DELETES THE SAMS:COMPRESS JCL LIBRARY
/*
/*
/******
//DEL      EXEC PGM=IEFB14
//JCL      DD  DISP=(OLD,DELETE),DSN=user.shrink.jcl
/*
/*
/******
/*
/* THIS STEP ALLOCATES A USER-DEFINED SAMS:COMPRESS JCL LIBRARY
/*
/*
/*
/******
//ALLOC    EXEC PGM=IEFB14
//JCL      DD  DISP=(NEW,CATLG),DSN=user.shrink.jcl,
//          SPACE=(TRK,(15,2,10)),
//          VOL=SER=volser,UNIT=unit
/*
/*
/******
/*
/* THIS STEP LOADS SHRINK.JCL TO THE USER-DEFINED JCL LIBRARY
/*
/*
/*
/******
//LOADPDS  EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//TAPE1    DD  DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=volser),
//          DSN=SHRINK.JCL
//DISK1    DD  DISP=OLD,DSN=user.shrink.jcl
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN    DD  *
//          COPY INDD=((TAPE1,R)),OUTDD=DISK1
/*
/*

```

Figure 2-1. JCL to Load the Installation JCL Data Set

Loading the SAMS:Compress Load Library

The first job, INSTALL1 (provided in the JCL Library), loads the SAMS:Compress load modules from tape to disk. The JCL for INSTALL1, shown in Figure 2-2, contains three steps:

- DEL deletes an existing SAMS:Compress load library data set.
- ALLOC allocates space for the new data set.
- LOADPDS loads the data set to disk.

```
/*INSTALL1 JOB
/*
/******
/*
/* THIS STEP DELETES THE SAMS:COMPRESS LOAD LIBRARY
/*
/******
//DEL EXEC PGM=IEFB14
//LOADLIB DD DISP=(OLD,DELETE),DSN=user.shrink.load
/*
/******
/* THIS STEP ALLOCATES A USER-DEFINED SAMS:COMPRESS LOAD LIBRARY
/*
/******
//ALLOC EXEC PGM=IEFB14
//LOADLIB DD DISP=(NEW,CATLG),DSN=user.shrink.load,
// SPACE=(TRK,(15,2,15)),
// VOL=SER=volser,UNIT=unit
/*
/******
/* THIS STEP LOADS SHRINK.LOAD TO THE USER-DEFINED LOAD LIBRARY
/*
/******
//LOADPDS EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE2 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=VOLSER),
// LABEL=(2,SL,,IN),DSN=SHRINK.LOAD
//DISK2 DD DISP=OLD,DSN=user.shrink.load
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN DD *
COPY INDD=((TAPE2,R)),OUTDD=DISK2
/*
/*
```

Figure 2-2. JCL to Load the Load Modules

Installing the Interactive Dialog

The SAMS:Compress Interactive Dialog is installed by means of a four-step process of loading the Interactive Dialog libraries and merging them with your existing libraries. The PDS libraries (GENLIB and FDTLIB) are allocated. Next the Interactive Dialog Control File is initialized and a primary TSO user is defined. Then

selected ISPF members are modified. Once the last step is completed, you can log on and define the authorized users for the Interactive Dialog.

If you have both SAMS:Compress for IMS and SAMS:Compress for MVS installed, please note that you must have separate Dialog Control Files and ISPF libraries for the Dialog component of each. This is necessary in order to maintain independent release levels for each product.

Note: The control word ENQ has been changed to RESERVE throughout the Interactive Dialog JCL procedures. This change prevents multiple CPU updates from causing problems with the Interactive Dialog Control File. Users in installations which use a global shared resource control product (MSX, GRS, etc.), must make the following modifications to these products: add the QNAME, “SHRINKFE”; add the RNAME of the data set name that was chosen for the Interactive Dialog Control File. If these changes are not made, reserve contention problems (“deadly embrace”) may result.

System Upgrading Considerations for SAMS:Compress 5.2

These paragraphs provide information for SAMS:Compress users who use the services of the SAMS:Compress Interactive Dialog and who are upgrading from SHRINK Release 4.5 (Dialog Release 1.2) to SAMS:Compress 5.2 (Dialog Release 5.2). Users who are upgrading from a release prior to SHRINK Release 4.5 should simply establish a new Dialog Control File.

To upgrade from SHRINK Release 5.1 (Dialog Release 1.3) to SAMS:Compress 5.2 (Dialog Release 5.2), no Dialog Control File Conversion is required. However, to upgrade from SHRINK Release 4.5 (Dialog Release 1.2) to SAMS:Compress 5.2 requires a conversion of the Dialog Control File. This is accomplished using a conversion program called SFEPFC13). The JCL to execute this program is located in the SHRINK.JCL data set on your installation tape. The member name is SFEPFC13. Before you execute the Control File Reformat program, you must modify the JCL as follows:

1. Replace the data set name on line 8 with the data set name for your new Interactive Dialog ISPLLIB.
2. Replace the data set name on line 9 with the data set name for your current Interactive Dialog Control File.

To prevent ABENDS, the reformat program initializes all of the new fields to zero or blank.

Note: Those data bases and files that were compressed in the old release of the Interactive Dialog Control File are based on old DASD space recommendations and costs. In order to produce new DASD space, DASD costs and performance information, you must extract these data bases and files with the new release. If you choose to do this, you should copy and archive the old Control File under a new name according to your naming conventions. Then perform extraction for the data bases or files in the new Control File, and complete Byte Distribution Analysis and Test Compression to obtain the new information.

Loading the Interactive Dialog Libraries

The next job you should run is INSTAL20. This job loads the libraries from tape to disk.

```

/* INSTAL20 JOB
/*
/******
/*
/* THIS STEP ALLOCATES THE USER-DEFINED SAMS:COMPRESS FOR IMS      *
/* INTERACTIVE DIALOG LIBRARIES                                     *
/******
//ALLOC EXEC PGM=IEPBR14
//ISPF DD DISP=(NEW,CATLG),DSN=user.shrims.clist,
// SPACE=(TRK,(3,1,5)),
// VOL=SER=volser,UNIT=unit
//ISPF DD DISP=(NEW,CATLG),DSN=user.shrims.ispllib,
// SPACE=(TRK,(80,5,15)),
// VOL=SER=volser,UNIT=unit
//ISPF DD DISP=(NEW,CATLG),DSN=user.shrims.ispplib,
// SPACE=(TRK,(60,5,80)),
// VOL=SER=volser,UNIT=unit
//ISPF DD DISP=(NEW,CATLG),DSN=user.shrims.ispslib,
// SPACE=(TRK,(15,2,25)),
// VOL=SER=volser,UNIT=unit
//ISPF DD DISP=(NEW,CATLG),DSN=user.shrims.ispmlib,
// SPACE=(TRK,(10,2,25)),
// VOL=SER=volser,UNIT=unit
/*
/******
/*
/* THIS STEP LOADS THE SHRINK.ISPF.XXXX DATA SETS TO THE        *
/* USER-DEFINED LIBRARIES                                       *
/******
//LOADPDS EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE8 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=VOLSER),
// LABEL=(8,SL,,IN),DSN=SHRINK.ISPF.CLIST
//TAPE9 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,REF=*.TAPE8),
// LABEL=(9,SL,,IN),DSN=SHRINK.ISPF.ISPLLIB
//TAPE10 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,REF=*.TAPE8),
// LABEL=(10,SL,,IN),DSN=SHRINK.ISPF.ISPPLIB
//TAPE11 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,REF=*.TAPE8),
// LABEL=(11,SL,,IN),DSN=SHRINK.ISPF.ISPSLIB
//TAPE12 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,REF=*.TAPE8),
// LABEL=(12,SL,,IN),DSN=SHRINK.ISPF.ISPMLIB
//DISK8 DD DISP=OLD,DSN=user.shrims.clist
//DISK9 DD DISP=OLD,DSN=user.shrims.ispllib
//DISK10 DD DISP=OLD,DSN=user.shrims.ispplib
//DISK11 DD DISP=OLD,DSN=user.shrims.ispslib
//DISK12 DD DISP=OLD,DSN=user.shrims.ispmlib
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN DD *
COPY INDD=((TAPE8,R)),OUTDD=DISK8
COPY INDD=((TAPE9,R)),OUTDD=DISK9
COPY INDD=((TAPE10,R)),OUTDD=DISK10
COPY INDD=((TAPE11,R)),OUTDD=DISK11
COPY INDD=((TAPE12,R)),OUTDD=DISK12
/*
/*

```

Figure 2-3. JCL to Load the Dialog Libraries

Allocating GENLIB and FDTLIB Libraries

You must then create certain libraries for GENLIB and FDTLIB that are required. These libraries are allocated by running the INSTAL21 (Figure 2-4) member of the JCL library after modifying the data set names to conform to your installation's requirements. The libraries are the following:

1. The GENLIB is a PDS library with a RECFM of FB and a LRECL of 80. The BLKSIZE is determined by the installer. This data set will contain all generated JCL and access method control blocks (DBDs, Clusters, etc.) generated by the Interactive Dialog.
2. The FDTLIB is a normal PDS load library whose DCB information should conform to the standards within your environment.

```

/*INSTAL21 JOB
/*
/******
/*
/* THIS JOB ALLOCATES THE USER-DEFINED SAMS:COMPRESS FOR IMS
/* GENLIB AND FDTLIB DATA SETS
/*
/******
/*
//ALLOC EXEC PGM=IEFBR14
/*
//GENLIB DD DISP=(NEW,CATLG),DSN=user.shrims.genlib,
//          SPACE=(TRK,(40,5,30)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=blksize),
//          VOL=SER=volser,UNIT=unit
/*
//FDTLIB DD DISP=(NEW,CATLG),DSN=user.shrims.fdtlib,
//          SPACE=(TRK,(40,5,30)),
//          DCB=(RECFM=U,BLKSIZE=blksize),
//          VOL=SER=volser,UNIT=unit
/*
/*

```

Figure 2-4. Allocating GENLIB and FDTLIB

Initializing the Interactive Dialog Control File

After the data sets have been loaded to DASD, you must allocate and initialize the Interactive Dialog Control File. This process consists of:

- Modifying member SFECNTL in the JCL library.
- Modifying member INSTAL22 in the JCL library.
- Executing the INSTAL22 job.

Member SFECNTL, shown in Figure 2-5, contains the IDCAMS control statements to define the Interactive Dialog Control file. You must modify these control statements to conform to your installation's requirements. First change the NAME fields to conform to your installation's data set naming conventions. The TRACKS fields

must be modified to provide the proper amount of space that will be needed. This is determined by the user. The space required is dependent upon the number of data bases/segments which are to be recorded and the number of FDTs that will be generated.

```
DELETE (user.shrimms.control.file)
DEFINE CLUSTER -
    (NAME(user.shrimms.control.file) -
    INDEXED -
    SHAREOPTIONS(4 3) -
    VOLUMES(VOLSER)) -
    DATA -
    (NAME(user.shrimms.control.file.DATA) -
    CYLINDERS(04 1) -
    CONTROLINTERVALSIZE(4096) -
    FREESPACE(20 10) -
    KEYS(56 0) -
    RECORDSIZE(350 1024))-
    INDEX -
    (NAME(user.shrimms.control.file.INDEX) -
    TRACKS(4 2) -
    CONTROLINTERVALSIZE(1024) -
    IMBED)
```

Figure 2-5. Control Statements for VSAM Cluster Def.

deftdasd represents the Default DASD device type.

name represents the name of the installation site. The company name is limited to 50 characters including spaces.

The format of the second input control statement:

A is a constant, and indicates this is the Authority code.

tsoid represents the TSO logon ID of the primary authorized user to install and implement the Interactive Dialog.

lastname, firstname mi represents the name of the primary authorized user which includes last name, first name, and middle initial. The comma should be included to separate the last name with the first name, but this is only cosmetic and will not cause an error if left missing. This field may be up to 25 characters long.

username represents the authorized user also. This is the name the Interactive Dialog will address during processing. This name is limited to 15 characters and can represent user name or user code developed by your installation.

User Modifications to the Interactive Dialog Libraries

Before you can logon to the Interactive Dialog, some modifications must be made to conform to your organization's standard naming conventions. The following libraries will need to be modified:

1. ISPPLIB Library.
2. ISPSLIB Library.
3. CLIST Library.

The instructions for modifying these libraries begin below.

1. Modifying the ISPPLIB.
 - a. Modify member SFEP\$UUI in the ISPPLIB library. This panel must be modified to contain the name or position title, or both, of the person that has been assigned the responsibility of identifying authorized users of the product.

```

SAMS:Compress for IMS -- INTERACTIVE DIALOG -- UNKNOWN USER ID -----
COMMAND ===

                                     DATE === 94/01/10
                                     TIME === 16:02

BECAUSE OF INTERNAL TRACKING, ALL USERS OF THE SAMS:Compress
INTERACTIVE DIALOG MUST BE DEFINED TO THE SYSTEM AND TSO
USERID _____ IS NOT CURRENTLY DEFINED.

IF YOU FEEL THAT YOU NEED TO USE THIS FACILITY TO DEFINE AND/OR
MAINTAIN SAMS:Compress CONTROL DATA, PLEASE CONTACT THE FOLLOWING
PERSON AND EXPLAIN YOUR POSITION AND NEEDS:

THE USER
      MUST ENTER
                THE NAME OF
                                THE PERSON TO
                                                CONTACT IN THIS SPACE.

PRESS THE 'ENTER' KEY OR THE 'END' PFKEY TO RETURN TO THE
SAMS:Compress INTERACTIVE DIALOG MANAGER AND TERMINATE THE CURRENT
DIALOG SESSION.

```

Figure 2-7. Panel SFEP\$UUI to Define TSO Users

2. Modifying the ISPSLIB.
 - a. The SFESSORT skeleton (Figure 2-8) is used to generate the SORT JCL that is appended to the data sample extract JCL. (The function of this JCL is described in the SAMS:Compress chapter, in the section discussing implementation of the product.) You must supply the correct data set name for the SORTLIB DD statement. If the SORTLIB data set is part of your installation's LINKLST, then delete the SORTLIB DD statement in this skeleton JCL procedure.

```

/**
/**          ***** SHRINK.DIAL.IPSLIB( SFESSORT ) *****
/** *****
/** SORT THE EXTRACTED DATA FOR PROCESSING BY "SFEPBDA" AND "SFECMPR"
/** *****
//SFESSORT EXEC PGM=SORT,REGION=1024K
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSIN DD *
)IM SFEEEXSCD NT
/*
//SORTMSG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SORTDIAG DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*,HOLD=YES
//SORTIN DD DSN=%%EXTRACT,DISP=(OLD,DELETE)
//SORTOUT DD DSN=&SFEEEXTDS,
)SEL &SFEEEXMED EQ T
//
// DISP=(NEW,CATLG,DELETE),UNIT=&SFEEDEV,
)ENDSEL
)SEL &SFEEEXMED EQ D
//
// DISP=(NEW,CATLG,DELETE),UNIT=&SFEEDEV,
//
// SPACE=(&SFEEUA,(&SFEEPA,&SFESA),RLSE,,ROUND),
)ENDSEL
//
// DCB=(RECFM=VBS,LRECL=32738,BLKSIZE=6233)

```

Figure 2-8. Extract JCL for Better BDA and Comp. Tests

- b. Skeleton member SFEEEXSCD (see Figure 2-9) contains the SORT control statements necessary for the SORT JCL in skeleton SFESSORT. The SORT control statements are for use with DFSORT. If your processing environment uses a sort facility other than DFSORT, you may find it advantageous to update skeleton SFEEEXSCD.

```

SORT FIELDS=(5,24,BI,A)
RECORD TYPE=V
OPTION MAINSIZE=512K,MSGDDN=SORTMSG

```

Figure 2-9. Sort Control Statements (SFEEEXSCD)

- c. Skeleton SFEVSAMP (Figure 2-10) is used within data sample extract JCL generated for an IMS data base using the BATCH DL/I batch method. The skeleton SFEVSAMP defines the VSAM buffer pool. If necessary, modify this skeleton to conform to your VSAM buffer pool requirements.

```

24576,8
4096,8
2048,8
1024,8

```

Figure 2-10. VSAM Buffer Pool Control Statements

3. Modifying the CLIST Library. Modify member SHRIMS, Figure 2-11, in the CLIST library as follows:
 - a. The CFDSN variable string “user.shrims.control.file” must be changed to reflect the actual name of the Interactive Dialog Control File.
 - b. The “user.shrims.genlib” must be changed to reflect the actual name of the GENLIB you allocated.
 - c. The FDTLIB variable string “user.shrims.fdtlib” must be changed to reflect the actual name of the FDTLIB you allocated.
 - d. The LLIB variable string “user.shrims.isplib” must be changed to reflect the actual name of the data set which contains the Interactive Dialog load modules.
 - e. Replace the MLIB variable string “user.shrims.isplib” with the name of the data set that contains the Interactive Dialog ISPF messages.
 - f. Replace the PLIB variable string “user.shrims.isplib” with the name of the data set that contains the Interactive Dialog ISPF panels.
 - g. Replace the SLIB variable string “user.shrims.isplib” with the name of the data set that contains the Interactive Dialog ISPF skeletons.
 - h. The SOUT variable “a” specifies the default SYSOUT class. The default is present each time the Interactive Dialog is executed. The SYSOUT class can be modified while the Interactive Dialog is active, but reverts back to the default for the next execution (LOGOFF/LOGON). You may choose a valid default for your installation site.
 - i. The Interactive Dialog supports the ISPF LIBDEF facility. If you have the ISPF Release 2.2 with full LIBDEF support, skip the next section, concatenating the Interactive Dialog Libraries, and go on to the section entitled “Activating the Interactive Dialog.” If you do not have the ISPF Release 2.2 with full LIBDEF support, delete all “ISPEXEC LIBDEF” statements from the SHRIMS CLIST member and continue to the next section, “Concatenating the Interactive Dialog Libraries.”

```
PROC 0 CFDSN(user.shrims.control.file) +
    GENLIB(user.shrims.genlib) +
    FDTLIB(user.shrims.fdtlib) +
    LLIB(user.shrims.ispllib) +
    PLIB(user.shrims.ispplib) +
    MLIB(user.shrims.ispmlib) +
    SLIB(user.shrims.ispslib) +
    SOUT(a)

CONTROL NOLIST NOMSG NOCONLIST
ISPEXEC LIBDEF ISPLLIB
FREE FILE(SFELLIB)
ALLOC DDNAME(SFELLIB) DA('&LLIB') SHR
ISPEXEC LIBDEF ISPLLIB LIBRARY ID(SFELLIB)
ISPEXEC LIBDEF ISPMLIB DATASET ID('&MLIB')
ISPEXEC LIBDEF ISPPLIB DATASET ID('&PLIB')
ISPEXEC LIBDEF ISPSLIB DATASET ID('&SLIB')
ISPEXEC SETMSG MSG(SFEM$000)
ISPEXEC CONTROL DISPLAY LOCK
ISPEXEC DISPLAY PANEL(SFEP$000)
FREE FILE(SFECNTL SFEREPT SFESOUT ISPPFILE SFESRCE SFESNAP)
FREE FILE(SFEIDCIN SFEIDCPT)
FREE FILE(SFELIN SFEUT1 SFELIB SFELMOD SFELINK)
FREE FILE(DBDLIB PSBLIB ACBLIB)
ISPEXEC SETMSG MSG(SFEM$001)
ISPEXEC CONTROL DISPLAY LOCK
ISPEXEC DISPLAY PANEL(SFEP$000)
SET &SFECFDSN = &CFDSN
SET &SFEGENDS = &GENLIB
SET &SFEDTDS = &FDTLIB
SET &SFEMVSDS = &MVSLIB
SET &SFESFEDS = &LLIB
ISPEXEC VPUT (SFECFDSN SFEGENDS SFEDTDS) SHARED
ISPEXEC VGET (SFESOCLS)
SET &COLON = :
IF &COLON = &COLON&SFESOCLS THEN &SFESOCLS = &SOUT
ISPEXEC VPUT (SFESOCLS) PROFILE
ALLOC DDNAME(SFECNTL) DA('&CFDSN') SHR
ALLOC DDNAME(ISPPFILE) DA('&GENLIB') SHR
ALLOC DDNAME(SFEIDCIN) UNIT(SYSDA) NEW SPACE(5) TRACKS
ALLOC DDNAME(SFEIDCPT) UNIT(SYSDA) NEW SPACE(5) TRACKS
ALLOC DDNAME(SFELIN) UNIT(SYSDA) NEW SPACE(5) TRACKS
ALLOC DDNAME(SFEUT1) UNIT(SYSDA) NEW SPACE(5) TRACKS
ALLOC DDNAME(SFELIB) DA('&FDTLIB') SHR
ALLOC DDNAME(SFELMOD) DA('&FDTLIB') SHR
ISPEXEC SELECT PGM(SFEPMAIN)
ISPEXEC SETMSG MSG(SFEM$998)
ISPEXEC CONTROL DISPLAY LOCK
ISPEXEC DISPLAY PANEL(SFEP$000)
FREE FILE(SFECNTL SFEREPT SFESOUT ISPPFILE SFESRCE SFESNAP)
FREE FILE(SFELIN SFEUT1 SFELIB SFELMOD SFELINK)
FREE FILE(SFEIDCIN SFEIDCPT)
FREE FILE(DBDLIB PSBLIB ACBLIB)
ISPEXEC LIBDEF ISPLLIB
ISPEXEC LIBDEF ISPMLIB
ISPEXEC LIBDEF ISPPLIB
ISPEXEC LIBDEF ISPSLIB
FREE FILE(SFELLIB)
END
```

Figure 2-11. CLIST to Initialize the Dialog (SHRIMS)

Concatenating the Interactive Dialog Libraries

Before activating the Interactive Dialog you must connect the Interactive Dialog's libraries to your TSO session. It may be helpful first to determine how your installation allocates the ISPF data sets, referred to by the ddnames listed below, to the TSO session.

DDNAME = SYSPROC	CLIST LIBRARY
DDNAME = ISPLLIB or ISPLOAD	PROGRAM LIBRARY
DDNAME = ISPPLIB	PANEL LIBRARY
DDNAME = ISPSLIB	SKELETON LIBRARY
DDNAME = ISPMLIB	MESSAGE LIBRARY

You can allocate the ISPF data sets in several ways. Figures 2-12 and 2-13 illustrate two primary methods for completing the allocation.

Figure 2-12 shows how to concatenate the Interactive Dialog libraries as part of your TSO LOGON proc.

```
//STEPLIB DD DISP=SHR,DSN=SYS1.ISPLOAD
//          DD DISP=SHR,DSN=user.shrims.ispllib
//SYSPROC DD DISP=SHR,DSN=SYS1.SYSPROC
//          DD DISP=SHR,DSN=user.shrims.clist
//ISPPLIB DD DISP=SHR,DSN=ISP.R1M0.ISPPLIB
//          DD DISP=SHR,DSN=user.shrims.ispplib
//ISPSLIB DD DISP=SHR,DSN=ISP.R1M0.ISPSLIB
//          DD DISP=SHR,DSN=user.shrims.ispslib
//ISPMLIB DD DISP=SHR,DSN=ISP.R1M0.ISPMLIB
//          DD DISP=SHR,DSN=user.shrims.ispmlib
```

Figure 2-12. Allocating ISPF Data Sets, TSO LOGON proc

Figure 2-13 shows a CLIST that will concatenate the Interactive Dialog libraries with your ISPF libraries.

```
ALLOCATE FILE(ISPLLIB) DA('SYS1.ISPLOAD' +
                          'user.shrims.ispllib') SHR
ALLOCATE FILE(SYSPROC) DA('SYS1.SYSPROC' +
                          'user.shrims.clist') SHR
ALLOCATE FILE(ISPPLIB) DA('ISP.R1M0.ISPPLIB' +
                          'user.shrims.ispplib') SHR
ALLOCATE FILE(ISPSLIB) DA('ISP.R1M0.ISPSLIB' +
                          'user.shrims.ispslib') SHR
ALLOCATE FILE(ISPMLIB) DA('ISP.R1M0.ISPMLIB' +
                          'user.shrims.ispmlib') SHR
```

Figure 2-13. Allocating ISPF Data Sets, CLIST

Which ever method you choose, it is important that you check the blocksize of the libraries you are concatenating. The blocksize of the first library must be greater than or equal to the succeeding library or libraries. Another way to prevent an error is to provide a large BLKSIZE for the first data set of the concatenation. Then either include a DCB parameter in your TSO LOGON proc **OR** define an attribute in the CLIST with a large BLKSIZE and allocate the data sets using the following attribute:

```
//ISPPLIB DD DISP=SHR,DSN=ISP.R1M0.ISPPLIB,  
//          DCB=BLKSIZE=6400  
//          DD DISP=SHR,DSN=user.shrims.ispplib  
  
or,  
  
ATTRIB DSATTR BLKSIZE(6400)  
ALLOCATE FILE(ISPPLIB) USING(DSATTR) DA('ISP.R1M0.ISPPLIB' +  
          'user.shrims.ispplib') SHR
```

Activating the Interactive Dialog

The second step is the actual activation of the Interactive Dialog, which can be accomplished by executing the SHRIMS CLIST. Either issue the TSO command or provide the execution of the CLIST as a menu selection option. The primary authorized user, established earlier by the execution of the INSTAL22 job, must activate the system. When the SAMS:Compress PRIMARY SERVICES MENU is displayed, select Option 9 to perform control maintenance. When the CONTROL MAINTENANCE MENU is displayed, the following must occur.

1. Select Option 1. When the Maintain Installation Control Parameters Screen appears, verify and/or modify the following fields:

INSTALLATION NAME—The name used on all Interactive Dialog reports and in various dialog panels.

SECURED SAMS:COMPRESS SERVICES—Indicates whether or not the SAMS:Compress services are to be protected from unauthorized use.

SAMS:COMPRESS LEVEL—Indicates whether you have installed the full (“F”) SAMS:Compress product, or the Express (“E”) SAMS:Compress product.

DEFAULT DASD UNIT—The normal default DASD unit is SYSDA. The value in this field can be changed when the control file is initialized. It can also be modified on the panel. The value will be the unit specification used for any DASD device when work files are allocated or JCL is generated.

DEFAULT TAPE UNIT—The normal tape unit is TAPE. The default value can be changed on this panel.

CURRENT DFP LEVEL—This field has no initial default value. It must be specified during the dialog initialization process, and must be changed whenever the installation upgrades its DFP level. This

value must be current, as it affects the space and cost calculations performed by the test compression program.

CURRENCY UNIT—Enter the currency type that you wish to appear in your costing reports (dollars, pounds, etc.). Enter the name of the currency unit, not the symbol.

DASD PURCHASE COST—Enter the amount that you spend to acquire new DASD devices, calculated for one megabyte. Enter the amount to the hundredth of the currency unit.

DASD CHARGE/MO—Enter the amount that your company charges its users for the use of one megabyte of DASD space for one month. Specify the amount in hundredths of the currency unit.

PRIME BATCH CPU CHARGE—Enter the amount that your company charges for one hour of CPU time to run batch jobs during prime time.

PRIME ON-LINE CPU CHARGE—Enter the amount that your company charges for one hour of CPU time to run online applications during prime time hours.

PRIME TIME EXCP CHARGE—Enter the amount that your company charges for one thousand EXCPs during prime time.

NIGHT BATCH CPU CHARGE—Enter the amount that your company charges for one hour of CPU time to run batch jobs during non-prime time hours.

NIGHT ON-LINE CPU CHARGE—Enter the amount that your company charges for one hour of CPU time to run online applications during non-prime time hours.

NIGHT TIME EXCP CHARGE—Enter the amount that your company charges for one thousand EXCPs during non-prime time hours.

PROCESSOR SPEED—Enter the speed of one of the processors in your CPU. The processor speed is entered in millions of instructions per second. For example, if you have a 32 MIP CPU and the CPU has two processors, the processor speed would be 16. If you have a 32 MIP CPU and the CPU has four processors, the processor speed would be 8.

2. Select Option 2. When the **USER MAINTENANCE MENU** appears, select Option C, enter your own TSO ID and press <Enter>. When the **USER DATA MAINTENANCE** screen appears, indicate which **SAMS:Compress Interactive Dialog Services** you are using by placing a “Y” in the appropriate fields and pressing <Enter>. Exit the User Maintenance Menu to return to the Control Maintenance Menu.
3. Select Option 3. The **MAINTAIN DEFAULT DATA SET NAME** screen is displayed. This screen has two sections, both of which must be completed.

After these steps have been performed, the Interactive Dialog is ready for use by the authorized users.

Authorizing a User for the Interactive Dialog

Each user of the Interactive Dialog must be identified as an authorized user to the system. To add a user to the authorization list, go to the SAMS:Compress PRIMARY SERVICES MENU and select Option 9, Maintain SAMS:Compress Interactive Dialog Control Data. When the CONTROL MAINTENANCE MENU is displayed, select Option 2. When the USER MAINTENANCE MENU is displayed, enter the new user's TSO ID, then enter an "A" for the type of maintenance desired and press <Enter>.

When the USER DATA MAINTENANCE screen is displayed, enter the name of the new user (last, first, middle initial) in the FULL NAME field. Then enter the name by which the user prefers to be called in the MESSAGE NAME field. Place the letter Y after every basic service that the user is to be authorized to use. When the maintenance is complete, press <Enter> to record the changes.

Updating the User Data Information in the Control File

From the CONTROL MAINTENANCE MENU, select Option 2. When the USER MAINTENANCE MENU is displayed, select Option C, enter the TSO ID of the user whose data is to be updated and press <Enter>.

When the user's data is displayed, make the desired updates and press <Enter> to record the changes.

Deleting an Authorized User From the Control File

From the CONTROL MAINTENANCE MENU, select Option 2. When the USER MAINTENANCE MENU is displayed, enter the user's TSO ID, then enter a maintenance code of "D" and press <Enter>. When the user's identity information is displayed, verify that it is the correct user to delete. If it is, enter a Y into the DELETE CONFIRM CODE field if it is the correct user and press <Enter>. If the displayed user identity is not for the correct user, press <Enter> and the user deletion will be canceled.

Listing the Interactive Dialog Authorized Users

To determine what users in your installation have been identified to the SAMS:Compress Interactive Dialog as authorized, go to the CONTROL MAINTENANCE MENU and select Option L. A list of authorized users will appear on your screen. You can use your FORWARD/DOWN, BACKWARD/UP, and LEFT and RIGHT PFkeys to browse the list if it is longer than can be displayed on a single screen.

To produce a printed copy of this list, press your <End> PF key and provide the appropriate information.

Installing SAMS:Compress for IMS

If you are installing SAMS:Compress, you will use JCL for INSTALL7, INSTALL8, and INSTALL9.

Loading the SAMS:Compress for IMS Library

INSTALL7 (Figure 2-14) loads the SAMS:Compress for IMS source modules from tape to disk.

```

/*INSTALL7 JOB
/*
/******
/*
/* THIS STEP DELETES THE SAMS:COMPRESS FOR IMS SOURCE LIBRARY *
/*
/******
//DEL      EXEC PGM=IEFBRL4
//IMS      DD  DISP=(OLD,DELETE),DSN=user.shrink.ims.source
/*
/******
/*
/* THIS STEP ALLOCATES A USER-DEFINED SAMS:COMPRESS FOR IMS *
/* SOURCE LIBRARY *
/*
/******
//ALLOC    EXEC PGM=IEFBRL4
//IMS      DD  DISP=(NEW,CATLG),DSN=user.shrink.ims.source,
//          SPACE=(TRK,(40,2,20)),
//          VOL=SER=volser,UNIT=unit
/*
/******
/*
/* THIS STEP LOADS SHRINK.IMS.SOURCE TO THE USER-DEFINED *
/* SAMS:COMPRESS FOR IMS SOURCE LIBRARY *
/*
/******
//LOADPDS EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//TAPE5    DD  DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=VOLSER),
//          LABEL=(5,SL,,IN),DSN=SHRINK.IMS.SOURCE
//DISK5    DD  DISP=OLD,DSN=user.shrink.ims.source
//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN    DD  *
//          COPY INDD=((TAPE5,R)),OUTDD=DISK5
/*
/*

```

Figure 2-14. JCL to Load the SAMS:Compress Source Lib.

Setting the Active File Descriptor Table Parameter

Note: The active file descriptor parameter may be set at the time of the original installation, or may be performed at a later time, depending on your installation's needs.

Each segment type to be compressed has an associated File Descriptor Table (FDT). Often FDTs are shared between segment types. If the maximum number of

different FDTs in use during the period in which the system is up will ever exceed 350, this being the default maximum, you must modify the SHRINKCB CSECT. Otherwise, IMS will terminate with a user abend and the message:

IMSHRINK. TOO MANY FDTs. REASSEMBLY REQUIRED.

To increase the maximum of 350 to 500, or any other specified amount, use the JCL and control cards in INSTALL8, shown in Figure 2-15, to execute the IBM IEBUPDTE utility. Then reassemble and relink SHRINKCB with the NORENT option. CSA usage can be tuned by reducing this value. Each increment uses 72 bytes of CSA. Be sure to allow for future FDTs.

```

/*INSTALL8 JOB
/*
/******
/*
/* SET THE ACTIVE FILE DESCRIPTOR TABLE (FDT) PARAMETER VALUE IN *
/* THE SHRINKCB SOURCE MODULE. *
/* *
/* SEE THE DESCRIPTION OF THIS STEP IN THE INSTALLATION PROCEDURES *
/* CHAPTER FOR COMPLETE DETAILS ON THIS PARAMETER VALUE. *
/* *
/*          =====> DO NOT RENUMBER THIS MEMBER <===== *
/* *
/******
/*
//FDTPARM EXEC PGM=IEBUPDTE
//SYSUT1 DD DISP=SHR,DSN=user.shrink.ims.source
//SYSUT2 DD DISP=SHR,DSN=user.shrink.ims.source
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
./ CHANGE NAME=SHRINKCB
SHRMAXN EQU 350 MAX NUMBER OF DIFFERENT ACTIVE FDT'S 00220000
/*
/*

```

Figure 2-15. JCL to Set FDT Value in SHRINKCB

Generating the SAMS:Compress Program Modules

Next run INSTALL9. This job assembles and link-edits the source contained in SHRINK.IMS.SOURCE to create the program modules. The SAMS:Compress program modules use the IMS/VS macros: DFSLEV, ISCD, IDLI, IMODULE, and ISWITCH. To ensure valid macro expansions during the assemblies, use the IMS/VS macro library that relates to the IMS/VS release you are currently using. You must reassemble these modules whenever you change IMS/VS releases.

Note: SAMS:Compress for IMS release 5.2 now supports IMSHRINK residing above the 16M line. With IMS/ESA 3.1 or higher installed, you can take advantage of this feature by adding "RMODE=ANY" to the linkedit steps of INSTALL9. IMSPASS must remain AMODE=24 and RMODE=24.

The JCL for INSTALL9 (Figure 2-16) contains the procedures to:

1. Assemble and link-edit the IMSPASS utility module.

2. Assemble and link-edit the re-entrant IMSHINK module and the associated non-re-entrant control block module, SHRINKCB.

Note: You can safely ignore the resulting IEW0461 warning messages:

IMSHRINK-CPLP, CPLPUFDN, SHRKGFD, and SHRKLINK

```

/**INSTALL9 JOB
/**
/*****
/**
/** THE ASMIMS ASSEMBLY IN-STREAM PROCEDURE STARTS HERE
/**
/*****
//ASMIMS PROC N=
//ASM EXEC PGM=IEV90,PARM='NODECK,OBJECT,XREF(SHORT)',REGION=512K
//STEPLIB DD DISP=SHR,DSN=sys1.asmh.linklib {1}
//SYSLIB DD DISP=SHR,DSN=SYS1.MACLIB {2}
// DD DISP=SHR,DSN=imsvs.genlib {3}
// DD DISP=SHR,DSN=imsvs.genliba {3}
// DD DISP=SHR,DSN=imsvs.genlibb {3}
// DD DISP=SHR,DSN=user.shrink.ims.source {4}
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSLIN DD DISP=(MOD,PASS),DSN=&&SYSLIN,
// SPACE=(80,(200,500,2)),UNIT=SYSDA
//SYSIN DD DISP=SHR,DSN=user.shrink.ims.source(&N) {4}
// PEND
/**
/*****
/**
/** THE ASMIMS ASSEMBLY IN-STREAM PROCEDURE ENDS HERE
/**
/*****
/**
/*****
/** THE LKEDIMS LINKEDIT IN-STREAM PROCEDURE STARTS HERE
/**
/*****
//LKEDIMS PROC LKPARM=
//LKED EXEC PGM=IEWL,PARM='&LKPARM',COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSLMOD DD DISP=SHR,DSN=user.shrink.load {5}
//RESLIB DD DISP=SHR,DSN=imsvs.reslib {6}
//SYSLIN DD DISP=(OLD,DELETE),DSN=&&SYSLIN
// DD DDNAME=SYSIN
// PEND
/**
/*****
/**
/** THE LKEDIMS LINKEDIT IN-STREAM PROCEDURE ENDS HERE
/**
/*****
/**
/*****
/**
/** ASSEMBLE AND LINKEDIT IMSPASS UTILITY PROGRAM MODULE
/**
/** IF IMPLEMENTING SAMS:COMPRESS FOR IMS USING THE INTERACTIVE
/** DIALOG, SKIP THE NEXT TWO STEPS (ASM1 AND LKED1)
/**
/*****

```

Figure 2-16. JCL to Assemble and Link Modules

```

//ASM1 EXEC ASMIMS,N=IMSPASS
//LKED1 EXEC LKEDIMS,LKPARM='LET,LIST,NCAL,NORENT,XREF'
//LKED.SYSIN DD *
ENTRY IMSGET
REPLACE GET
INCLUDE SYSLMOD(SHRINK)
INCLUDE RESLIB(DFSLE000)
NAME IMPASS(R)
//*
//*
/*****
/* ASSEMBLE AND LINKEDIT THE IMSHRINK PROGRAM MODULES *
/*
/*****
//ASM2 EXEC ASMIMS,N=IMSHRINK
//ASM3 EXEC ASMIMS,N=SHRNKERR
//ASM4 EXEC ASMIMS,N=SHRKABND
//ASM5 EXEC ASMIMS,N=SHRKDLTE
//ASM6 EXEC ASMIMS,N=SHRKFREM
//ASM7 EXEC ASMIMS,N=SHRKGTEM
//ASM8 EXEC ASMIMS,N=SHRKLKNG
//ASM9 EXEC ASMIMS,N=SHRKLKLD
//ASM10 EXEC ASMIMS,N=SHRKTWO
//LKED2 EXEC LKEDIMS,LKPARM='LET,LIST,NCAL,RENT,XREF,AMODE=31'
//LKED.SYSIN DD *
ENTRY IMSHRINK
REPLACE SHRKGFDT EXPECT IEW0461 FOR SHRKGFDT
REPLACE SHRKLKNG EXPECT IEW0461 FOR SHRKLKNG
REPLACE CPLP EXPECT IEW0461 FOR CPLP
REPLACE CPLPUFDN EXPECT IEW0461 FOR CPLPUFDN
REPLACE SHRNKCVT
INCLUDE SYSLMOD(SHRKEXPD)
NAME IMSHRINK(R)
//*
/*****
/* ASSEMBLE AND LINKEDIT THE SHRINKCB PROGRAM MODULE. *
/*
/*****
//ASM11 EXEC ASMIMS,N=SHRINKCB
//LKED3 EXEC LKEDIMS,LKPARM='LET,LIST,NCAL,NORENT,XREF,AMODE=31'
//LKED.SYSIN DD *
NAME SHRINKCB(R)
//*
//

```

Figure 2-16. JCL to Assemble and Link Modules (cont.)

When you have completed these steps, the SAMS:Compress for IMS system is completely installed.

Notes for Figure 2-16.

1. The DSN of the Assembler H library.
2. The DSN of the standard system macro library.
3. The DSNs of the IMS/V S macro libraries. These libraries must contain the IMS/V S macros DFSLEV, ISCD, IDLI, IMODULE and ISWITCH, which must be compatible with the current IMS/V S release to ensure valid macro expansion.

4. The DSN assigned to SHRINK.IMS.SOURCE. It must be the same DSN specified on the DISK5 DD statement in the LOADPDS step of the INSTALL7 job.
5. The DSN assigned to SHRINK.LOAD. It must be the same DSN specified on the DISK2 DD statement in the LOADPDS step of the INSTALL1 job. The DSN of the installation load library where the linkage editor output will be placed. The DSN shown represents SHRINK.LOAD.
6. The DSN of the IMS/VS RESLIB. This library must contain the IMS DFSLI000 module.

System Upgrading Considerations

There was an error in the original coding of SHRINK Release 3.4 that could cause unkeyed segments to be compressed as keyed segments with a key length of 1. System Modifications 172, 175, and 184, which are incorporated in Release 3.5 and above, correct the above coding problem. Because of these System Modifications, an incompatibility may exist between SHRINK Release 3.4 and SHRINK Release 3.5. The incompatibility exists for your data base if the conditions listed below are true.

Your data base has unkeyed data segments that are compressed, **AND** one of the following conditions exists:

1. The SEGCC macro for the unkeyed segment was coded as FN=(fdtname,SHR).
2. The FDT named in the SEGCC macro on an unkeyed segment was used by another segment, **AND**
 - a. The other segment was coded as FN=(fdtname,SHR), **OR**
 - b. The key length plus the key offset of the other segment match its TYPE N RDL specifications.
3. SM172 was added to your SHRINK system, unkeyed segments were inserted and SM175 was added.

If the above conditions exist in your data base, unload the data base using your SHRINK Release 3.4 system and reload the data base using the current SAMS:Compress release.

Note: If running IMS/ESA release 4.1, SAMS:Compress for IMS 5.2.1 or higher is required.

Note: When you change an FDT and the data base is involved in online processing, bring the online system down and back up so that the new FDT may be loaded.

Online Environment

To access compressed records in an online environment, the library containing modules IMSHRINK, SHRINKCB, and the FDT load modules must be authorized. Include this library in the STEPLIB concatenation of the IMS JCL procedure.

Batch Processing

To access compressed records during batch processing, authorize and include in the concatenation, the same libraries listed for online environments. If not authorized, include in the procedure a DFSRESLB dd statement (pointing to IMSVS.RESLIB).

Installing SAMS:Compress for IMS Express

Loading the Express Library

Note: Modify the install JCL, INSTAL17, //SYSLIB and point to the correct MACLIBs, GENLIBs, and RESLIB. INSTAL16, Figure 2-17, loads the Express source modules from tape to disk.

```
/*INSTAL16 JOB
/*
/******
/*
/* THIS STEP DELETES THE SAMS:COMPRESS FOR IMS/EXPRESS          *
/* SOURCE LIBRARY                                              *
/******
//DEL      EXEC PGM=IEFB14
//IMS      DD DISP=(OLD,DELETE),DSN=user.shrink.imexp.source
/*
/******
/*
/* THIS STEP ALLOCATES A USER-DEFINED SAMS:COMPRESS FOR IMS/EXPRESS*
/* SOURCE LIBRARY                                              *
/******
//ALLOC    EXEC PGM=IEFB14
//IMS      DD DISP=(NEW,CATLG),DSN=user.shrink.imexp.source,
//          SPACE=(TRK,(40,2,20)),
//          VOL=SER=volser,UNIT=unit
/*
/******
/*
/* THIS STEP LOADS SHRINK.IMEXP.SOURCE TO THE USER-DEFINED    *
/* SAMS:COMPRESS FOR IMS/EXPRESS SOURCE LIBRARY                *
/******
//LOADPDS  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE13   DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=VOLSER),
//          LABEL=(13,SL,,IN),DSN=SHRINK.IMEXP.SOURCE
//DISK13   DD DISP=OLD,DSN=user.shrink.imexp.source
//SYSUT3   DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4   DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN    DD *
//          COPY INDD=((TAPE13,R)),OUTDD=DISK13
/*
/*
```

Figure 2-17. JCL to Load the Express Data Set

Generating the Express Program Modules

Next run INSTAL17. This job assembles and link-edits the source contained in SHRINK.IMEXP.SOURCE to create the program modules. The JCL for INSTAL17 is shown in Figure 2-18. It contains the procedures to assemble and link-edit the re-entrant IMEXPRES module. When you have completed this step, the Express system is completely installed.

```

//*INSTAL17 JOB
//*
//*****
//*
//* THE ASMIMS ASSEMBLY IN-STREAM PROCEDURE STARTS HERE *
//*
//*****
//ASMIMS PROC N=
//ASM EXEC PGM=IEV90,PARM='NODECK,OBJECT,XREF(SHORT)',REGION=512K
//STEPLIB DD DISP=SHR,DSN=sys1.asmh.linklib {1}
//SYSLIB DD DISP=SHR,DSN=SYSL.MACLIB {2}
// DD DISP=SHR,DSN=imsvs.genlib {3}
// DD DISP=SHR,DSN=imsvs.genliba {3}
// DD DISP=SHR,DSN=imsvs.genlibb {3}
// DD DISP=SHR,DSN=user.shrink.imexp.source {4}
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(300,50))
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=1089
//SYSLIN DD DISP=(MOD,PASS),DSN=&&SYSLIN,
// SPACE=(80,(200,500,2)),UNIT=SYSDA
//SYSIN DD DISP=SHR,DSN=user.shrink.imexp.source(&N) {4}
// PEND
//*
//*****
//*
//* THE ASMIMS ASSEMBLY IN-STREAM PROCEDURE ENDS HERE *
//*
//*****
//*
//* THE LKEDIMS LINKEDIT IN-STREAM PROCEDURE STARTS HERE *
//*
//*****
//LKEDIMS PROC LKPARM=
//LKED EXEC PGM=IEWL,PARM='&LKPARM',COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSLMOD DD DISP=SHR,DSN=user.shrink.load {5}
//RESLIB DD DISP=SHR,DSN=imsvs.reslib {6}
//SYSLIN DD DISP=(OLD,DELETE),DSN=&&SYSLIN
// DD DDNAME=SYSIN
// PEND
//*
//*****
//*
//* THE LKEDIMS LINKEDIT IN-STREAM PROCEDURE ENDS HERE *
//*
//*****
//*

```

Figure 2-18. JCL to Assemble and Link IMEXPRES

```

//*****
//*
//* ASSEMBLE AND LINKEDIT THE IMEXPRES PROGRAM MODULES
//*
//*****
//ASM1 EXEC ASMIMS,N=IMEXPRES
//ASM2 EXEC ASMIMS,N=SHRNKERR
//ASM3 EXEC ASMIMS,N=SHRKABND
//ASM4 EXEC ASMIMS,N=SHRKFREM
//ASM5 EXEC ASMIMS,N=SHRKGTEM
//ASM6 EXEC ASMIMS,N=SHRKWTO
//LKED1 EXEC LKEDIMS,LKPARM='LET,LIST,NCAL,RENT,XREF,AMODE=31'
//LKED.SYSIN DD *
ENTRY IMEXPRES
INCLUDE SYSLMOD(SHRKQIKI) BASE@023 IMXP@230 IMXP@523
NAME IMEXPRES(R)
//*
```

Figure 2-18. JCL to Assemble and Link IMEXPRES (cont.)

Notes for Figure 2-18.

1. The DSN of the Assembler H library.
2. The DSN of the standard system macro library.
3. The DSNs of the IMS/V S macro libraries. These libraries must contain the IMS/V S macros DFSLEV, ISCD, IDLI, IMODULE, and ISWITCH, which must be compatible with the current IMS/V S release to ensure valid macro expansion.
4. The DSN assigned to SHRINK.IMEXP.SOURCE. It must be the same DSN specified on the DISK13 DD statement in the LOADPDS step of the INSTAL16 job.
5. The DSN assigned to SHRINK.LOAD. It must be the same DSN specified on the DISK2 DD statement in the LOADPDS step of the INSTALL1 job. The DSN of the installation load library where the linkage editor output will be placed. The DSN shown represents SHRINK.LOAD.
6. The DSN of the IMS/V S RESLIB. This library must contain the IMS DFSLI000 module.

Note: The Express program modules use the IMS/V S macros: DFSLEV, ISCD, IDLI, IMODULE, and ISWITCH. To ensure valid macro expansions during the assemblies, use the IMS/V S macro library that relates to the IMS/V S release you are currently using. You must reassemble the Express modules whenever you change IMS/V S releases.

Note: SAMS:Compress for IMS release 5.2 now supports IMEXPRES residing above the 16M line. With IMS/ESA 3.1 or higher installed, you can take advantage of this feature by adding "RMODE=ANY" to the linkedit steps of INSTALL17.

Online Environments

To access compressed records in an online environment, the library containing the Express module IMEXPRES must be authorized. Include this library in the STEPLIB concatenation of the IMS JCL procedure.

Batch Processing

To access compressed records during batch processing, the same libraries as stated for online environments should be authorized and included in the STEPLIB concatenation. If not authorized, include a DFSRESLB dd statement (pointing to IMSVS.RESLIB) in your procedure.

Installing Support for IMS Fastpath Unload/Reload Utilities (Optional)

Job INSTALL23, provided in the JCL Library, loads the SAMS:Compress Fastpath utility support modules from tape to disk. The JCL for INSTALL23, shown in Figure 2-19, contains three steps:

- DEL deletes an existing load library data set.
- ALLOC allocates space for the new data set.
- LOADPDS loads the data set to disk.

```
/*INSTALL23 JOB
/*
/******
/* THIS STEP DELETES THE SAMS:COMPRESS FOR IMS *
/* FASTPATH UNLOAD/RELOAD (NON-IMS ENVIRONMENT) UTILITY SUPPORT *
/* LOAD LIBRARY *
/******
//DEL EXEC PGM=IEFB14
//IMS DD DISP=(OLD,DELETE),DSN=USER.SHRINK.IMSUT.LOAD
/*
/******
/* THIS STEP ALLOCATES A USER-DEFINED SAMS:COMPRESS FOR IMS *
/* FASTPATH UNLOAD/RELOAD (NON-IMS ENVIRONMENT) UTILITY SUPPORT *
/* LOAD LIBRARY *
/******
//ALLOC EXEC PGM=IEFB14
//IMS DD DISP=(NEW,CATLG),DSN=USER.SHRINK.IMSUT.LOAD,
// SPACE=(TRK,(40,2,20)),
// VOL=SER=VOLSER,UNIT=UNIT
/*
/******
/* THIS STEP LOADS SHRINK.IMSUT.LOAD TO THE USER-DEFINED *
/* SAMS:COMPRESS FOR IMS FASTPATH UNLOAD/RELOAD *
/* (NON-IMS ENVIRONMENT) UTILITY SUPPORT LOAD LIBRARY *
/******
//LOADPDS EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//TAPE14 DD DISP=OLD,UNIT=TAPE,VOL=(,RETAIN,SER=VOLSER),
// LABEL=(14,SL,,IN),DSN=SHRINK.IMSUT.LOAD
//DISK14 DD DISP=OLD,DSN=USER.SHRINK.IMSUT.LOAD
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN DD *
COPY INDD=((TAPE14,R)),OUTDD=DISK14
/*
```

Figure 2-19. JCL to Load the Load Modules

Once loaded, you must concatenate this library to the STEPLIB for any jobs which run Fastpath Unload/Reload batch utilities requiring the SAMS:Compress compression routines. Note: Those modules cannot be used in place of those used in the normal IMS environment.

Installing the Callable Subroutines for Use With CICS (Optional)

Complete the steps in this section if you wish to call the SAMS:Compress subroutines from programs running under CICS. This is not necessary to be able to use SAMS:Compress for IMS for data sets within a CICS region.

Loading the Source Modules

If you are installing SAMS:Compress to interface with CICS, the next job you should run is INSTAL13, shown in Figure 2-20. INSTAL13 loads the source modules from tape to disk. If you are not going to interface SAMS:Compress with CICS, skip this section.

```

/*INSTAL13 JOB
/*
/*
/******
/*
/* THIS STEP DELETES THE SHRINK SOURCE LIBRARY
/*
/******
//DEL      EXEC PGM=IEFB14
//SHR2     DD  DISP=(OLD,DELETE),DSN=user.shrink.shr2.source
/*
/*
/******
/*
/* THIS STEP PRE-ALLOCATES A USER-DEFINED SAMS:COMPRESS SOURCE
/* LIBRARY
/******
//ALLOC    EXEC PGM=IEFB14
//SHR2     DD  DISP=(NEW,CATLG),DSN=user.shrink.shr2.source,
//          SPACE=(TRK,(50,5,10)),
//          VOL=SER=volser,UNIT=unit
/*
/*
/******
/*
/* THIS STEP LOADS SHRINK.SHR2.SOURCE TO THE USER-DEFINED
/* SAMS:COMPRESS SOURCE LIBRARY
/*
/******
//LOADPDS EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//TAPE3     DD  DISP=OLD,UNIT=unit,VOL=(,RETAIN,SER=volser),
//          LABEL=(3,SL,,IN),DSN=SHRINK.SHR2.SOURCE
//DISK3     DD  DISP=OLD,DSN=user.shrink.shr2.source
//SYSUT3    DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSUT4    DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSIN     DD  *
//          COPY INDD=((TAPE3,R)),OUTDD=DISK3
/*
/*

```

Figure 2-20. JCL to Load the Callable Subroutines

Specify the FDT Names and a Fullword in the CWA

The INSTAL14 JCL, Figure 2-21, lets you specify:

- The offset from the start of the CSA of an aligned fullword in the CWA to be used exclusively by SAMS:Compress.
- The names of the FDTs for the data sets used via SAMS:Compress.

```
/* INSTAL14 JOB
/*
/*
/* *****
/*
/* SPECIFY THE OFFSET OF THE ALIGNED SHRINK FULLWORD IN THE CWA *
/* BY REPLACING SHRKWORD. THE OFFSET IS FROM THE START OF THE CSA *
/* (THE CWA IMMEDIATELY FOLLOWS THE CSA). *
/* *
/* SPECIFY THE NAMES OF THE FDTs BY REPLACING FDTNAME. *
/* *
/* *****
/* SPECIFY EXEC PGM=IEBUPDTE
/*SYSUT1 DD DISP=OLD,DSN=user.shrink.shr2.source
/*SYSUT2 DD DISP=OLD,DSN=user.shrink.shr2.source
/*SYSPRINT DD SYSOUT=*
/*SYSIN DD *
./ REPL NAME=SHRKWORD
CWASHRK EQU X'2D0'
./ REPL NAME=FDTNAMES
SCB mastfdt {1}
SCB tranfdt
SCB testfdt
/*
/*
```

Figure 2-21. JCL to Set the CWA Fullword Offset and FDT

Note for Figure 2-21.

1. Names assigned to FDTs that will be loaded into storage at CICS startup time.

Use the EQU and SCB statements in the figure only as examples. If you need to change the offset of the reserved fullword, repeat this process, the assembly, and link steps (shown in INSTAL15). Then relink the application programs.

For files that are used infrequently, you can omit the record in FDTNAMES that specifies the corresponding FDT. If you do, the FDT is loaded dynamically as needed, along with about 3K of storage for each transaction using that FDT. This storage, and the dynamically loaded FDT, is freed when it is no longer needed. Since each FDT can tie up from 4K to over 8K of storage, you can save space in this way or, whenever appropriate, use a single FDT for several files.

Generating the SAMS:Compress Program Modules

Next, run INSTAL15, which assembles and links the program modules. It contains two in-stream procedures, shown in Figure 2-22:

ASMSHR2—assembles the program modules.

LKEDSHR2—links the program modules.

```

/*INSTAL15 JOB
/*
/*
/******
/*
/* THE ASMSHR2 ASSEMBLY IN-STREAM PROCEDURE STARTS HERE
/*
/******
//ASMSHR2 PROC N=
//ASM EXEC PGM=IEV90,PARM='NODECK,OBJECT,XREF(SHORT)',REGION=512K
//STEPLIB DD DISP=SHR,DSN=sys1.asmh.linklib {1}
//SYSLIB DD DISP=SHR,DSN=SYS1.MACLIB {2}
// DD DISP=SHR,DSN=user.shrink.shr2.source, {3}
// DCB=SYS1.MACLIB
// DD DISP=SHR,DSN=cics.maclib {4}
// DD DISP=SHR,DSN=cics.source {5}
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT2 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSUT3 DD UNIT=SYSDA,SPACE=(1700,(400,400))
//SYSPRINT DD SYSOUT=*
//SYSLIN DD DISP=(MOD,PASS),DSN=&&SYSLIN,
// UNIT=SYSDA,SPACE=(800,(100,20))
//SYSIN DD DISP=SHR,DSN=user.shrink.shr2.source(&N) {3}
// PEND
/*
/******
/* THE ASMSHR2 ASSEMBLY IN-STREAM PROCEDURE ENDS HERE
/*
/******
/*
/* THE LKEDSHR2 LINKEDIT IN-STREAM PROCEDURE STARTS HERE
/*
/******
//LKEDSHR2 PROC LKPARM='LET,LIST,XREF,NCAL' {6}
//LKED EXEC PGM=IEWL,PARM='&LKPARM',COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSLMOD DD DISP=SHR,DSN=user.shrink.load {7}
//SYSLIN DD DISP=(OLD,DELETE),DSN=&&SYSLIN
// DD DDNAME=SYSIN
// PEND
/*
/*
/******
/* THE LKEDSHR2 LINKEDIT IN-STREAM PROCEDURE ENDS HERE
/*
/******

```

Figure 2-22. JCL to Assemble and Link Source Modules

```

//*****
//*
//* CREATE THE STARTUP MODULE SHRKNMOD.
//* REPEAT THESE STEPS IF CWASHRK CHANGES.
//*
//*****
//ASM1 EXEC ASMSHR2,N=SHTART
//ASM2 EXEC ASMSHR2,N=SHRKFREM
//ASM3 EXEC ASMSHR2,N=SHRKGETM
//ASM4 EXEC ASMSHR2,N=SHRKDLTE
//ASM5 EXEC ASMSHR2,N=SHRKL0AD
//ASM6 EXEC ASMSHR2,N=SHRKL0KG
//LKED1 EXEC LKEDSHR2
//LKED.SYSIN DD *
INCLUDE SYSLMOD(SHRKEXPD)
ENTRY SHTART
NAME SHRKNMOD(R)
//*
//*****
//*
//* CREATE THE STUB MODULE SHRKCICS.
//* REPEAT THESE STEPS IF CWASHRK CHANGES.
//*
//*****
//ASM7 EXEC ASMSHR2,N=SHRKCICS
//LKED2 EXEC LKEDSHR2
//LKED.SYSIN DD *
NAME SHRKCICS(R)
//*
//*
//*****
//*
//* CREATE SHRKSCBS. REPEAT THIS STEP IF EITHER CWASHRK OR
//* FDTNAMES CHANGES.
//*
//*****
//*
//ASM8 EXEC ASMSHR2,N=SHRKSCBS
//LKED3 EXEC LKEDSHR2
//LKED.SYSIN DD *
NAME SHRKSCBS(R)
//*

```

Figure 2-22. JCL to Assemble and Link Source Modules (cont.)

Notes for Figure 2-22.

1. The DSN of the Assembler H library.
2. The DSN of the standard system macro library.
3. The DSN assigned to SHRINK.SHR2.SOURCE. Refer to the LOADPDS step of the INSTAL13 job.
4. The DSN of the CICS macro library.
5. The DSN of the CICS source library.
6. Link-edit parameters must default or be specified for “AMODE=24” and “RMODE=24”.
7. The DSN assigned to SHRINK.LOAD. Refer to the LOADPDS step of the INSTALL1 job.

Assembling and Linking the Program Modules

Figure 2-22 shows the remaining steps in `INSTAL15` to assemble and link the `SHRNKMOD`, `SHRKCICS`, and `SHRKSCBS` modules. If you need to change `FDTNAMES`, the member containing the list of FDT names, you must assemble and link the `SHRKSCBS` module and update the PPT to reflect the current list of FDT names.

CICS and Application Program Requirements

After you run `INSTAL13`, you must define the `SAMS:Compress` elements in CICS tables and link the application programs as follows:

- In the Processing Program Table (PPT), define `SHRNKMOD`, `SHRKSCBS`, and all FDT modules to be used via `SAMS:Compress`. Specify all PPT entries as `PGMLANG=ASSEMBLER`, `PGMSTAT=ENABLED`, and `RELOAD=NO`.
- In the Program Load Table (PLT) that specifies programs to be executed during CICS initialization, define `SHRNKMOD`. Make sure that this PLT is identified by the `PLTPI` operand of the `DFHSIT` macro.
- Link-edit the application programs with the `SHRKCICS` module (for CICS, do not use `SHRKSTUB` or `SHRKEXPD` as specified in this chapter).
- Define the application programs in the PPT as usual.

Chapter 3. Implementation Considerations

Overview

SAMS:Compress for IMS provides data compression and expansion facilities for IMS/VS users. The program operates under control of the IBM Operating System OS/VS2 or MVS without application program modification. In an MVS environment, SAMS:Compress for IMS may be used with the IMS Parallel DL/I mode of data base control.

SAMS:Compress is implemented through the facilities of the Interactive Dialog, an ISPF interface. This chapter describes this method of implementing and operating SAMS:Compress. It is directed to the IMS data base administrator. The first part of the chapter provides a step-by-step procedural description of the implementation process, and considerations to use if you want to increase the compression by fine-tuning.

Note: If your facility does not use ISPF version 2.0 or higher, you must implement SAMS:Compress using a set of supplied utilities. This method of implementation is documented in the Batch SAMS:Compress Express chapter.

Features

SAMS:Compress is implemented as an IMS/VS Edit/Compression User Exit and is completely transparent to IMS application programs while providing:

- Compression for both fixed-length and variable-length segments.
- Data compression only; key sequence fields remain uncompressed.
- Compression of one or more segment types of a physical data base.

Guidelines for Choosing Data Bases to Compress

Not every data base is a good candidate for compression. The following guidelines should be considered when choosing data bases for compression:

1. The primary accessing of the data base should be random as is the normal case in an online operation.
2. A data base whose primary access is sequential should carefully be considered, because the run time of the program may increase to what could be considered unacceptable.
3. A data base is not compressed as a whole. It is compressed on a segment-by-segment basis. Segments containing fewer than 20 bytes of compressible area may cost more in compression and expansion overhead than the amount of DASD space saved would justify.

It is recommended that you choose one data base to compress first, rather than trying to compress several at one time. After you have been through the procedure a time or two, you will know better how to determine what effect any fine-tuning would have on compression results, and can make use of the services available to tailor the Record Definition Language, or share FDTs amongst segments or data bases. But for the first-time user, we suggest you use the default values generated by SAMS:Compress for most steps. It is recommended that your data sample contain 10% of the total occurrences of each segment you will be compressing.

Implementation for an IMS Data Base

Once the SAMS:Compress product has been installed, the Interactive Dialog initialized, and users authorized to the system, you can implement SAMS:Compress to compress a data base.

Implementation Steps Outline

Implementing SAMS:Compress for a data base involves a two-phase procedure. During the first phase, you will perform several tasks to identify the data base to the Interactive Dialog. During the second phase, you will compress the data base by performing several tasks, also using services provided through the Interactive Dialog.

Here is a summary of the tasks necessary. A detailed step-by-step description of each task follows this summary.

Identify an IMS Data Base to SAMS:Compress

- Analyze the Data Base
- Generate a Stored Data Report
- Specify Criteria for the Data Sample Extract

- Identify the Data Base for the Sample Extract
- Generate JCL for Testing
- Test the Effectiveness of the RDL on the Extract Sample

Compress an IMS Data Base with SAMS:Compress

- Identify the Data Base to SAMS:Compress
- Generate the FDTs for the Data Base
- Generate the DBD macros for the Data Base
- Assemble the DBD macros for the Data Base
- Unload the Data Base specifying the old DBD
- Move the FDTs and DBD macros into the Production Library
- Reload the Data Base using the New DBD Macros and FDTs. The actual compression is performed during the reload procedure.

Identify an IMS Data Base to SAMS:Compress

Analyze the Data Base

The data base analysis is performed as follows:

1. From the Interactive Dialog Primary Services Menu, select Option 1, SAMS:Compress SUPPORT.
2. When the SAMS:Compress SUPPORT SERVICES menu is displayed, select Option 1, Analyze a Data Base. The Data Base Analysis screen will then be displayed.
3. Provide the name of the DBD to be identified, the name of the DBDLIB that contains the DBD load module, and the title of the IMS data base. The standard DBDLIB name or the name of the last DBDLIB that you used will be displayed initially. If the DBDLIB name is the same, you can simply cursor down to the data base title field.
4. Provide a data base title. Because the coded name of a data base may not be familiar to everyone in your installation, you should provide a meaningful title for the data base.
5. When the required data has been entered, press <ENTER>. The Interactive Dialog will then obtain and analyze the DBD load module.
6. When the analysis is complete, a message will appear in the upper right-hand corner of the screen.

A variety of errors can occur during the analysis. If an error should occur, an error message will appear in the upper right-hand corner of your screen. Sometimes this initial error message is rather cryptic due to length limitations. If the meaning of the message is not clear, press your <HELP> PF key and a more complete message will be

displayed. If the more complete message requires more space than is available on one screen to describe the problem, a second message will direct you to press your <HELP> PF key again to display subsequent screens to fully describe the error.

Generate a Stored Data Report

As the analysis is performed, the information about the data base is stored in the Interactive Dialog Control File. To review the results of the analysis, follow these steps:

1. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 2, Generate a Data Base Report.
2. Enter the DBD name of the IMS data base on which the Stored Data Report is to be based, then press <ENTER>. The Stored Data Report will be generated and displayed on your screen.
3. Browse the contents by using the FORWARD/DOWN, BACKWARD/UP and the LEFT and RIGHT PF keys.
4. Press your <END> PF key to display the HARDCOPY screen. This screen will have a default value of 0 displayed, which will suppress printing of the Stored Data Report.
5. Enter a 1 in the print/suppress field, to produce a printed copy of the Stored Data Report, then enter the SYSOUT class to which the report is to be written.
6. Enter the number of copies that you want printed. The default is 1 copy, but you can specify up to 9 copies
7. After you have entered the desired hard copy controls, press <ENTER>. The Stored Data Report will then be copied to the desired SYSOUT print class. A message will be displayed in the upper right-hand corner of your screen indicating whether the report was printed successfully or not.

The Stored Data Report includes the following information:

For the entire data base:

- organization type (HDAM, HIDAM, HISAM, DEDB)
- number of data set groups
- number of levels
- number of segments
- number of second indexes
- names of other associated DBDs

For each segment of the data base:

- segment name

- segment level
- segment hierarchical position
- minimum and maximum length
- prefix length
- key field
- key location
- maximum compressible length
- current RDL statement
- FDT data
- data extract parameters
- byte distribution analysis
- best compression test results

Control information regarding the data base:

- DDNAME
- device type
- DBD name
- data set name

Basic testing information:

- description of JCL generated for this data base
- date and time of byte distribution analysis
- date and time of compression testing

A sample Stored Data Report for an IMS data base is shown in the Reports chapter.

Specify Criteria for the Data Sample Extract

Before extracting a data sample for testing, you must specify the criteria to be used to obtain the sample. You may specify where in the data base the sampling is to begin and end, and the quantity of occurrences to select for the sample. These criteria are specified through the USER PARAMETER MAINTENANCE MENU as follows:

1. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 3, USER PARAMETER MAINTENANCE.
2. From the User Parameter Maintenance Screen, select Option 2, Sample Data Extract Control Parameters.

The Extract Maintenance Control Screen displays the root segment name. The values displayed are default values, unless the segment has had criteria specified previously. The default value for these parameters is "0" for BYPASS, "ALL" for EXTRACT and "0" for SKIP.

3. The BYPASS parameter controls where the extraction will begin. To begin the extraction of the data sample with the first occurrence of the segment, use the default value of 0 for the BYPASS parameter.

To begin the extraction of the data sample at some point other than the first occurrence of the segment, enter the number of segments to bypass before beginning the extraction.

Note: The number you enter for the BYPASS parameter represents a multiple of 1,000 occurrences of the segment. If you enter a value of 10, the extraction will begin after 10,000 occurrences of the segment have been bypassed.

4. The EXTRACT parameter controls where the extraction will end. This value also represents multiples of 1,000 segments. The default value of ALL means that the extraction will continue until the end of data base is reached. If you enter a value of 15, the extraction will stop after 15,000 occurrences of the segment have been extracted.
5. The SKIP parameter controls the quantity of segments to be skipped between each segment extracted. The default of 0 means that once extraction has begun, every occurrence of the segment will be extracted until the extraction ends. To extract only half of the segments, enter a value of 1 for the SKIP parameter. The extraction will then select one segment, skip a segment, select the third segment, and so on. To extract only 10 percent of the segments, enter a value of 9 for the SKIP parameter. To select only 2 percent of the segments, enter a value of 49 for the SKIP parameter. Note that these percentage values are based on the segments between the beginning and ending boundaries specified for the extract. Only if you use the default values for the BYPASS and EXTRACT parameters will the percentage reflect the entire data base. Increasing the value of the SKIP parameter causes the quantity of segments extracted to decrease. Enter the appropriate value for the SKIP parameter.
6. After you have entered the desired changes, press your <END> PF key to return to the User Parameter Maintenance Menu.

Identify the Data Base for the Sample Extract

You extract a sample from the data base to provide a basis for SAMS:Compress to generate default values for compression processing.

A data sample is extracted from the data base to obtain fixed information which can be compressed, using compression controls. Testing with the same data permits valid comparison of the results of the test compressions. The data sample should contain 10,000 occurrences of a segment. When a segment has fewer than 10,000 occurrences, extract all occurrences. When a segment has more than 50,000 occurrences, you may limit the extraction to 20 percent of the occurrences and still obtain a valid sample.

From the SAMS:Compress SUPPORT SERVICES MENU, select Option 3 to display the USER PARAMETER MAINTENANCE MENU. When this screen is displayed, select Option 5, Data Set Name(s) to display a list of DDNAMEs.

When the DBD was first analyzed, all of the DBDs that were either directly or indirectly associated with the current DBD were examined and all of the required DDNAMEs were recorded. These are the DDNAMEs displayed. To extract the sample data directly from the data base using either DLI or HSSR calls, enter the correct data set name after each DDNAME.

You can browse the list of DDNAMEs using your FORWARD/DOWN and BACKWARD/UP PF keys. You may enter one or more data set names before pressing the <ENTER> key. After all data set names have been entered, press your <END> PF key to return to the USER PARAMETER MAINTENANCE MENU.

Generate JCL for Testing

The JCL procedures for the data sample extraction, byte distribution, and test compression processes are generated as described below:

1. From the SAMS:Compress Support Services Menu, select Option 4, Generate JCL for Testing.
2. When the JCL Generation Menu is displayed, enter the DBD name.
3. When the next JCL Generation screen is displayed, provide the following information:
 - a. Jobcard (once only).
 - b. The seven-character prefix for two JCL members that will be generated. The first member is the extract JCL. The second member will contain the byte distribution and the test compression JCL. Generating the JCL for both members is done only once for each extraction selection.

(You may choose to rerun byte distribution and test compression after modifying the RDL to obtain better compression results. In this case, resubmit the JCL from the second member only.)

4. Now select one of the extract options.
5. A Data Extraction JCL Generation screen appears. Enter or verify the information required for each selection.

For option 1, Extract using Batch DLI calls, enter the following:

- PSB name and PCB number.
- Indicate from which library the PSB will be retrieved, PSBLIB or ACBLIB.
- Verify that the PSBLIB, ACBLIB, and RESLIB data set names are correct.

Note: PSB language must be COBOL or Assembler. PL/I is not supported.

Note: The PCB must have sensitivity to every segment in the data base.

For option 2, Extract using BMP DLI calls, enter

- PSB name and PCB number.
 - IMS Control Region ID.
 - Verify that the RESLIB data set name is correct.
6. A second Data Extraction JCL Generation screen appears. Enter the required allocation information for the data set that will contain the extracted sample.

Two JCL members are now generated.

7. Execute the extraction JCL, then the byte distribution and test compression JCL.

The results from these runs are printed automatically to your default SYSOUT class.

Test the Effectiveness of the RDL on the Extract Sample

Typical compression results using default RDL range from about 30 to 50 percent, and may vary even further, due to the nature of the specific data base. It is possible to fine-tune the RDL and make other adjustments to the SAMS:Compress controls to improve the compression, but it is recommended that this be done only after you have gained some experience with the product.

If your compression test results are in or near this range, and you find the total compression results acceptable, you may now compress your first data base. However, if the compression test results for this data base are unacceptable, you have two options. You may identify a different data base to SAMS:Compress and test compression on it, or you may refer to the Record Definition Language chapter for information regarding customizing the RDL and perform fine-tuning at this point to improve the compression. (More details on fine-tuning are also supplied in “Obtaining the Best Compression Possible for a Data Base” later in this chapter.) Again, however, it is recommended that you gain experience on a few data bases using the default values before you attempt any fine-tuning.

Note that all the results of the definition process to date are saved in the Interactive Dialog Control File, and you can fine-tune and implement compression on this data base later if you desire.

Now that you have completed the definition phase and have produced compression test results that are acceptable, you are now ready to actually compress the data base.

Compress an IMS Data Base With SAMS:Compress

Identify the Data Base to SAMS:Compress

1. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 5, Implement Data Base Under SAMS:Compress Controls.
2. When the DATA BASE IMPLEMENTATION MENU appears, enter the DBD of the data base.

Generate the FDTs for the Data Base

3. Select Option 1, Select Segments for Compression, from the Data Base Implementation Menu. When the Data Base Segment Name screen appears, it displays the FDT SHARE status (a default value of NO is displayed for each segment for a given data base), and an area in which comments may appear.
4. For each segment in the list that is to be compressed using an FDT, enter an “F” in the select field and press <ENTER>. Enter an “E” to flag that segment for Express compression. Enter a “C” to cancel any previous compression for a given segment.

Generate the DBD Macros for the Data Base

This procedure converts the current IMS macros used to define the data base.

1. Select Option 2, Generate Modified DBD Source Macros, from the DATA BASE IMPLEMENTATION MENU. When the DBD GENERATION screen appears, specify the information called for:

Source DBDLIB
Source DBD Name
SAMS:Compress DBD Name
Reorganized DBD Name (optional)

Note: The purpose of the optional Reorganized DBD Name macros is to permit the reorganization of the data base without having to expand and compress the segments.

2. When all the information has been supplied, press <ENTER>. The SEGM macro is modified, and SEGCC macros are created and inserted for each segment for which an FDT was generated (in the previous step). The modified DBD macros are stored in GENLIB.

The following four functions (“Assemble the DBD macros...” through “Reload the Data Base...”) must be performed using the services of the IMS product itself. These steps are not performed by the Interactive Dialog.

Assemble the DBD Macros for the Data Base

Note: The SEGCC macro must be accessible to assemble the new DBD. This can be accomplished by concatenating the SAMS:Compress source library to the SYSLIB DD statement of your DBD Assembly proc.

Unload the Data Base Specifying the Old DBD

Move the FDTs and DBD Macros into the Production Library.

1. Move the FDT modules and SAMS:Compress load modules into the IMSVS.RESLIB or into a separate library of your choice. If you do not place them in the IMSVS.RESLIB, you must modify the JCL of every job that accesses the data base. The modification will consist of concatenating the libraries that contain the SAMS:Compress FDTs and SAMS:Compress load modules to a STEPLIB or JOBLIB DD statement.
2. Move the new DBD module into the appropriate DBD libraries and perform an ACB Gen to do a DBD BUILD.

Reload the Data Base Using the New DBD Macros and FDTs.

Note: If the necessary FDTs are not accessible to the compressed data base, a 3997 ABEND will occur.

When the reorganization is complete, the data base will be under the control of the SAMS:Compress data compression product.

Obtaining the Best Compression for a Data Base

To determine the best possible compression you can achieve for a data base, you will need to use the facilities of the Interactive Dialog. You will also need the report from a SMUII of the data base or from a data base reorganization. This will assist you in determining which segments to compress. Follow these steps to determine the best compression obtainable for the data base:

1. Analyze the data base. Using the Interactive Dialog, select the SAMS:Compress Support option. When the SAMS:Compress PRIMARY SERVICES MENU is displayed, select Option 1, Analyze a Data Base. Complete the analysis process as described earlier in this chapter to determine the structure of the data base and to set the SAMS:Compress user parameter default values.
2. Generate a Stored Data Report. After the data base has been analyzed, generate a Stored Data Report by selecting Option 2 from the SAMS:Compress PRIMARY SERVICES MENU. Produce a hard copy of the report.
3. Specify the data sample extract parameters. Based upon the contents of the analysis report and the SMUII or reorganization report, specify the data sample extract parameters. The extract parameters should be specified based upon the following considerations:

- a. If a segment has fewer than 20 bytes of compressible data, specify the EXTRACT parameter with a value of 000.
 - b. If a segment has fewer than 10,000 occurrences, specify the EXTRACT parameter with a value of ALL.
 - c. If a segment has between 10,000 and 20,000 occurrences, extract at least 10,000 by specifying the EXTRACT parameter with a value of ALL, and the SKIP parameter with a value of 1. You may also extract ALL occurrences by leaving the SKIP parameter at its default value of 0.
 - d. If a segment has more than 20,000 occurrences, extract at least 10,000 occurrences and use the SKIP or BYPASS parameters to spread the extraction across the entire data base.
4. Identify where the data sample is found. Using Option 5 from the USER PARAMETER MAINTENANCE MENU, identify where the data sample will be found. To extract the data sample directly from the data base, supply the correct data set name for each entry in the DDNAME table.
 5. Generate the JCL for extraction and testing. Select Option 4, Generate JCL for Testing, from the SAMS:Compress PRIMARY SERVICES MENU.

Note: You can select Option 2 only if your installation has installed the HSSR facility.

6. Execute the JCL for testing. Note that the actual data base is not compressed by this procedure. Rather, the compression process is simulated on the extracted sample, and a Compression Test Report is generated indicating the compression percentage which can be expected, using the current compression controls for this data base.

You now have a basis on which you can judge the effectiveness of the RDL for each segment. Fine-tuning the RDL for a segment can increase compression, but may increase the CPU overhead.

You may respecify RDL for various segments and perform the testing procedure again. Note that if you change the RDL for a segment, the Byte Distribution Analysis (BDA) for the segment is invalidated and the procedure must be re-executed.

As you fine-tune the RDL for the segments in the data base, follow these steps:

1. Specify the RDL you have devised for each segment. Use Option 3, RDL User Parameters, of the USER PARAMETER MAINTENANCE MENU.
2. Specify any FDT sharing to be used. Use Option 4, FDT User Parameters, of the USER PARAMETER MAINTENANCE MENU. (See “Sharing an FDT Between Two or More Segments” later in this chapter.)
3. Execute the JCL to calculate byte distribution and perform test compression.
4. Evaluate the results of the test compression.

The above process can be repeated as often as needed until you have obtained the compression that you want to achieve for the data base.

When your testing and evaluating have produced the optimum compression controls, follow the step-by-step procedure outlined earlier in this chapter to compress the data base.

Considerations in Specifying Compression Controls

Sharing an FDT Between Two or More Segments

As a means of conserving resources, a single FDT can be used by multiple segments. In the shared FDT environment, the base segment will use its own RDL; the sharing segments will use the RDL of the base segment. Therefore, when you modify the RDL of a segment that is the base of a shared FDT, you will also be modifying the RDL of every segment that shares the FDT.

You can check the Stored Data Report for a data base to see if a segment serves as a base segment (shares the FDT of another segment). If the report indicates a segment shares another segment's FDT, the report will identify the FDT's base segment and the data base in which it is found. If the report indicates the segment is the base for a shared FDT, the report will list all of the segments sharing the FDT.

To specify a shared FDT, you will use the USER PARAMETER MAINTENANCE MENU. From the SAMS:Compress SUPPORT SERVICES menu, select Option 3. When the USER PARAMETER MAINTENANCE MENU is displayed, select Option 4, FDT User Parameters. A list of the FDT control parameters will be presented for each segment in the data base.

The following fields are used to specify whether an FDT is to be shared and, if so, how the sharing is implemented:

FDT name
SHR
SRCE DBD
SRCE SEG

The SHR field controls sharing of an FDT. For the normal (unshared) condition, the SHR field is specified with a value of NO. When SHR is NO, the FDT name should be the same as the segment name, the SRCE DBD should be the name of the current DBD, and the SRCE SEG should be the same as the segment name.

Two methods are available for FDT sharing. The first is when the segments have exactly the same field structure. The second is when the segments have the same basic characteristics, but the structure may be different.

Method 1—SHR=SFS

The first method of sharing an FDT may be used when two or more segments have exactly the same field structure, and is accomplished by specifying a value of SFS (same field structure) in the SHR field of both (or all) segments.

Select one of the segments to serve as the base segment for the FDT. Its SHR field will be specified with a value of “SFS”. Enter the name of the FDT to be shared in its FDT NAME field and its own segment name in its SRCE SEG field.

Normally an FDT is only shared between segments in the same data base. It is recommended that FDTs not be shared between data bases, but it is possible to share an FDT created for a segment in another data base. When base segment for the FDT is in another data base, enter the name of that DBD in the SRCE DBD field. Otherwise that field is left blank.

For the other segment(s), the SHR field will be specified with a value of “SFS”. The FDT NAME field should be specified with the same name as that specified in the base segment. Enter the name of the base segment in the SRCE SEG field. If the base segment is in another data base, enter the name of that DBD in the SRCE DBD field.

Method 2–SHARE=YES

The second method for sharing an FDT may be used when the segments have the same basic characteristics, but the structure may be different. This means the segments must have the same length non-compressible key area, and the remainder of the RDL for the segment is defined as C1VER. If these two conditions exist, the segments may share the RDL

Nx, C1VER

where x=the non-compressible key area. This method is accomplished by specifying a value of “YES” for the SHR field in both (or all) segments.

Select one of the segments to serve as the base segment for the FDT. The SHR field will be specified with a value of “YES”. For the base segment, the other two fields (SRCE DBD and SRCE SEG) must not be changed.

For the other segment(s), the SHR field will be specified with a value of “YES”. For the FDT NAME field, specify the name of the FDT this segment will share. For the SRCE SEG field, enter the name of the base segment.

Normally an FDT is only shared between segments in the same data base, but it is possible to share an FDT created for a segment in another data base. When base segment for the FDT is in another data base, enter the name of that DBD in the SRCE DBD field. Otherwise that field is left blank.

Entering or Updating the RDL for a Data Base Segment

Note: Do not change the RDL for a data base segment until you have verified that the segment does not share an FDT with another segment. (See “Sharing an FDT Between Two or More Segments” earlier in this chapter for further information.)

Once you have prepared custom RDL for a data base segment, you may enter the RDL through the USER PARAMETER MAINTENANCE MENU. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 3, User Parameters Maintenance. The USER PARAMETER MAINTENANCE MENU will be displayed.

If this is the first function you have selected for this session, the DBD name field will be blank. If other functions have been performed, this field will contain the name of the last DBD referenced. If the name is incorrect or the field empty, supply the correct DBD name, then press <ENTER>.

To enter or update the RDL, select Option 3, RDL User Parameters. Your screen will display a list of the compressible segments in the data base. You will also see the segment's key information and the number of compressible bytes the segment contains.

Move the cursor to the select field in front of the segment name and enter an "s" for "select." The current RDL for that segment will be displayed. Notice that the RDL for the segment key is protected and cannot be altered. The remaining RDL is for the compressible area.

The RDL for the compressible area may be entered. If the field displays default RDL, you may change it. You may use up to 11 lines of 78 bytes each for the segment's RDL. When all changes to the RDL have been made, press <ENTER> to store the new RDL. (If for some reason you decide to cancel the changes you have made in the RDL, press your <END> PF key, and you will be returned to the list of segments for this data base.)

If you have made any errors in specifying the RDL, an error message will be displayed in the upper right-hand corner of the screen and the cursor will appear at the location of the error. To obtain further information about the error, press your <HELP> PF key. Correct the error in the RDL and press <ENTER>.

When the RDL is correct, the list of segments will be displayed again. The select field in front of the segment will contain an asterisk (*) if the segment has been selected during this session. Note that segments can be selected multiple times for RDL maintenance in the same session. You may select another segment for RDL maintenance or end the RDL maintenance task.

If you choose to end the RDL maintenance for this session, press your <END> PF key while the segment list is displayed and you will be returned to the USER PARAMETER MAINTENANCE MENU.

Each time you run a compression test, SAMS:Compress builds a record of the compression achieved, and compares it to other test runs, and stores this information in the Control File. The Compression Test Report shows the RDL which obtained the compression, so it is important to retain these reports when you perform repeated testing. When the BEST TEST field on the Compression Test Report does not reflect the current test number, you may need to refer to the Compression Test Report for the best test to re-enter the RDL producing the best results for the segment.

You may easily determine if a data base has already been identified to SAMS:Compress and tested, and if so, what were the results. Select Option I from the SAMS:Compress SUPPORT SERVICES MENU. A list of the recorded data bases will be displayed on your screen. You may browse the list using your

FORWARD/DOWN and BACKWARD/UP PF keys. To produce a printed copy of the list, press your <END> PF key.

Controlling the Size of a Compressed Segment

You may control the size of a compressed segment either by use of the PAD facility or by specifying a minimum size for the segment, or both. The pad will be applied first and then checked against the MIN SIZE specification.

Specifying padding or a minimum size is accomplished through the USER PARAMETER MAINTENANCE MENU. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 3. When the USER PARAMETER MAINTENANCE MENU is displayed, select Option 4, FDT User Parameters. This will display a list of segments in the data base, along with the value for each segment's PAD LEN, PAD TYP and MIN SIZE fields.

The padding facility is used to tell SAMS:Compress to add padding to the segment after it has been compressed to the smallest possible size. This is done for segment insert only. This can be used to help reduce VSAM CI splits. Padding can be specified in one of two ways:

1. By adding a specific number of bytes to the compressed segment
2. By padding the compressed segment by a percentage of the number of bytes saved during compression

To specify a number of bytes to add, enter that number in the PAD LEN field. You can specify from 1 through 255 bytes. To specify a percentage of the compressed length, enter that number in the PAD LEN field. You can specify from 1 percent through 99 percent. In the PAD TYP field, enter an "N" for adding a specific number of bytes; enter a "P" for a percentage value.

$$xx,P = \text{Percentage, can be calculated as } A + ((B - A) * P) / 100,$$

where:

A = compressed segment length

B = uncompressed segment length

P = percentage specified

The minimum size facility permits you to specify that the SAMS:Compress routines will not compress the segment to a length less than the MIN SIZE value. While not required, MIN SIZE may be specified to reduce the probability of VSAM CI splits during replacement operations. A MIN SIZE specification can reduce the variation in length of compressed segments; during replacement operations, the segments are less likely to increase in length.

The default for the MIN SIZE field is 0 which tells SAMS:Compress to compress the segment to the smallest possible size. You may specify a value of 1 through 999 bytes for this field.

Move your cursor to the desired segment and enter the appropriate information under the PAD LEN and the PAD TYP fields, or the MIN SIZE field.

By using your FORWARD/DOWN and your BACKWARD/UP PF keys, you can browse the list of segments if it is greater than a single screen. When you are through specifying segment size values, press your <END> PF key and you will be returned to the User Parameter Maintenance Menu.

Deleting an IMS Data Base From the Control File

If, on the basis of the compression testing, you determine that compressing the data base is not desirable, there are several follow-up steps you may wish to take. You may simply leave the results of the testing in the Control File for reference purposes. But if you don't want to retain the test results, you can delete the information from the Control File.

To delete the information from the Control File, select Option 8, Delete Obsolete Data Base Information, from the SAMS:Compress SUPPORT SERVICES MENU. When the DELETE DATA BASE FROM SAMS:Compress CONTROL FILE screen appears, enter the DBD name of the data base that is to be deleted, then press <ENTER>.

As a safeguard against inadvertently deleting the wrong data base, you will be asked to confirm your entry on the SAMS:Compress DBD DATA DELETION CONFIRMATION screen. The DBD name, the data base title, and other pertinent information will be presented. The default value for the confirmation field is N, for no delete. Take a moment to verify that this is the correct data base to delete. If it is, enter a Y in the confirmation field and press <ENTER>. The information about the data base will then be erased from the control file.

If you entered the wrong DBD name and need to cancel the delete request, press <ENTER> or your <END> PF key.

If you delete information on a data base that has already been compressed under SAMS:Compress, you will not cause any harm. However, we recommend that you do not delete the test results for these data sets, but only delete information for data sets that are not and will not be compressed using SAMS:Compress.

Changing a Data Base Controlled by SAMS:Compress

If any changes are made to the structure of an IMS data base already compressed and under the control of SAMS:Compress, you must also adjust the compression control parameters. Depending on the nature of the changes, there are varying tasks which must be performed.

When the changes to the data base affect the SAMS:Compress-controlled segments or the physical structure of the data base changes, you must complete the following steps:

1. Generate a Stored Data Report of the information about the data base that is currently stored in the Control File.
2. Create an uncompressed test version of the data base in its new structure. This can be accomplished with a data base reorganization/restructure which used the normal SAMS:Compress DBD to unload the data and the new structure DBD without any SAMS:Compress controls to do the restructure/restore.
3. Delete the information about the data base from the Control File after the Stored Data Report has been generated.
4. Analyze the new data base structure as if it has never been processed previously.

Other kinds of changes to the data base require different actions. For the following kinds of changes, modify the SAMS:Compress version of the DBD for the macro.

1. For changes in the length of a segment under the control of SAMS:Compress, correct the bytes parameter in the SEGM macro for the segment. For variable length non-DEDB segments, the maximum length required by SAMS:Compress is 12 bytes larger than the defined length. For fixed length segments compressed with a variable RDL (i.e., C1VER.) or DEDB segments, the length remains the same.

Note: For fixed length segments compressed with a fixed RDL (i.e., C1Fxx.) the prior four steps must be executed to build a new FDT.

2. For changes to any of the segments not under SAMS:Compress, Regenerate the DBD using the Interactive Dialog DBD generation service.

Chapter 4. Implementing Express

Overview

SAMS:Compress for IMS Express provides data compression and expansion facilities for IMS/VS users. The program operates under control of the IBM Operating System OS/VS2 or MVS without application program modification. In an MVS environment, Express may be used with the IMS Parallel DL/I mode of data base control.

Express can be implemented through the facilities of the Interactive Dialog, an ISPF interface, or by a simple modification of the IMS DBD Source macros. A user wishing to implement Express by modifying the IMS DBD Source macros should refer to the “Batch SAMS:Compress” chapter.

This chapter describes the procedures for implementing and operating Express through the facilities of the Interactive Dialog. It is directed to the IMS data base administrator. The first part of the chapter provides a step-by-step procedural description of the implementation process. This is followed by illustrations of each Interactive Dialog panel used to implement Express, along with a description of how the user is to supply the required information for each field.

Note: If a user’s facility does not use ISPF version 2.0 or higher, Express must be implemented as documented in the “Batch SAMS:Compress” chapter.

The IMS services of the Interactive Dialog does perform compression simulation, therefore it is available to you for analysis purposes only. However, this Express chapter does not describe data compression simulation. If you wish to determine the effect of Express before actually compressing a data base, follow the instructions described in the “Implementation Considerations” chapter.

Express Features

Express is implemented as an IMS/VS Edit/Compression User Exit. The use of Express is completely transparent to IMS application programs and provides:

- Compression for both fixed-length and variable-length segments.
- Data compression only; key fields (sequence fields) remain uncompressed.
- Compression of one or more segment types of a physical data base.
- No JCL changes are required when using Express.

Implementing Express for an IMS Data Base

Once the SAMS:Compress product has been installed, the Interactive Dialog initialized, and users authorized to the system, you can implement Express to compress a data base.

Implementation Steps Summary

Implementing Express for a data base involves a two-phase procedure. During the first phase, you will perform several tasks to identify the data base to the Interactive Dialog. During the second phase, you will compress the data base by performing several tasks, also using services provided through the Interactive Dialog.

Here is a summary of the tasks necessary. A detailed step-by-step description of each task follows this summary.

Identify an IMS Data Base to Express

- Analyze the Data Base (This step is required)
- Generate a Stored Data Report (This step is optional)

Compress an IMS Data Base with Express

- Identify the Data Base to the Interactive Dialog.
- Generate the DBD macros for the Data Base.
- Assemble the DBD macros for the Data Base.
- Unload the Data Base specifying the old DBD.
- Move the DBD macros into the Production Library.
- Reload the Data Base using the New DBD Macros. The actual compression is performed during the reload procedure.

Identify an IMS Data Base to Express

Analyze the Data Base

1. From the Interactive Dialog Primary Services Menu, select Option 1, SAMS:Compress SUPPORT.
2. When the SAMS:Compress SUPPORT SERVICES MENU is displayed, select Option 1, Analyze a Data Base. The Data Base Analysis screen will then be displayed.
3. Provide the name of the DBD to be identified, the name of the DBDLIB that contains the DBD load module, and the title of the IMS data base. The standard DBDLIB name or the name of the last DBDLIB used will

be displayed initially. If the DBDLIB name is the same, a user can simply cursor down to the data base title field.

4. Provide a data base title. Because the coded name of a data base may not be familiar to everyone in the installation, a meaningful title for the data base should be provided.
5. When the required data has been entered, press <ENTER>. The Interactive Dialog will then obtain and analyze the DBD load module.
6. When the analysis is complete, a message will appear in the upper right-hand corner of the screen.

A variety of errors can occur during the analysis. If an error should occur, an error message will appear in the upper right-hand corner of the screen. Sometimes this initial error message is rather cryptic due to length limitations. If the meaning of the message is not clear, press the <HELP> PF key and a more complete message will be displayed. If the more complete message requires more space than is available on one screen to describe the problem, a second message will direct you to press the <HELP> PF key again to display subsequent screens to fully describe the error.

Generate a Stored Data Report

As the analysis is performed, the information about the data base is stored in the Control File. To review the results of the analysis, follow these steps:

1. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 2, Generate a Data Base Report.
2. Enter the DBD name of the IMS data base on which the Stored Data Report is to be based, then press <ENTER>. The Stored Data Report will be generated and displayed on your screen.
3. Browse the contents by using the FORWARD/DOWN, BACKWARD/UP and the LEFT and RIGHT PF keys.
4. Press the <END> PF key to display the HARDCOPY screen. This screen will have a default value of 0 displayed, which will suppress printing of the Stored Data Report.
5. Enter a 1 in the print/suppress field, to produce a printed copy of the Stored Data Report, then enter the SYSOUT class to which the report is to be written.
6. Enter the number of copies to print. The default is 1 copy, but a user can specify up to 9 copies.
7. After you have entered the desired hard copy controls, press <ENTER>. The Stored Data Report will then be copied to the desired SYSOUT print class. A message will be displayed in the upper right-hand corner of the screen indicating whether the report was printed successfully or not.

The Stored Data Report includes the following information:

For the entire data base:

- organization type (HDAM, HIDAM, HISAM, DEDB)
- number of data set groups

- number of levels
- number of segments
- number of second indexes
- names of other associated DBDs

For each segment of the data base:

- segment name
- segment level
- segment hierarchical position
- minimum and maximum length
- prefix length
- key field
- key location
- maximum compressible length
- current RDL statement
- FDT data
- data extract parameters
- byte distribution analysis
- best compression test results

Control information regarding the data base:

DDNAME

- device type
- DBD name
- data set name

Basic testing information:

- description of JCL generated for this data base

- date and time of byte distribution analysis
- date and time of compression testing

A sample Stored Data Report for an IMS data base is shown in the “Reports” chapter.

Once a user has completed the definition phase, he is now ready to actually compress the data base.

Compress an IMS Data Base With Express

Identify the Data Base to the Interactive Dialog

1. From the SAMS:Compress SUPPORT SERVICES MENU, select Option 5, Implement Data Base Under SAMS:Compress Controls.
2. When the DATA BASE IMPLEMENTATION MENU appears, enter the DBD of the data base.

Select Segments to be Compressed

Select Option 1, Select Segments for Compression, from the DATA BASE IMPLEMENTATION MENU. When the Segment Selection screen appears, it will display a list of the segments for this data base. For each segment in the list that is to be compressed, enter a “E” (for “express”) in the select field and press <ENTER>. The selected segment(s) will be marked for compression when the DBD is generated.

Generate the DBD Macros for the Data Base

This step converts the current IMS macros used to define the data base.

1. Select Option 2, Generate Modified DBD Source Macros, from the DATA BASE IMPLEMENTATION MENU. When the DBD Generation screen appears, specify the information called for:
 - Source DBDLIB
 - Source DBD Name
 - SAMS:Compress DBD Name
 - Reorganized DBD Name (optional)

Note: The purpose of the optional reorganized DBD name macros is to permit the reorganization of the data base without having to expand and compress the segments.

2. When all the information has been supplied, press <ENTER>. The SEGM macro is modified for each segment flagged in the previous step. The modified DBD macros are stored in GENLIB.

3. The following four functions must be performed using the services of the IMS product itself. They are not performed by the Interactive Dialog.

Assemble the DBD Macros for the Data Base

Unload the Data Base Specifying the Old DBD

Move the DBD Macros into the Production Library

Move the new DBD module into the appropriate DBD libraries and perform an ACB Gen to do a DBD BUILD.

Reload the Data Base Using the New DBD Macros

When the reorganization is complete, the selected segments of the data base will be under the control of Express.

Deleting an IMS Data Base From the Control File

If a user determines that compressing the data base is not desirable, there are two possible follow-up steps. A user may simply leave the data base information in the Control File for reference purposes. But if you do not want to retain the information, you can delete it from the Control File.

To delete the information from the Control File, select Option 8, Delete Obsolete Data Base Information, from the SAMS:Compress SUPPORT SERVICES MENU. When the DELETE DATA BASE FROM SAMS:Compress CONTROL FILE screen appears, enter the DBD name of the data base that is to be deleted, then press <ENTER>.

As a safeguard against inadvertently deleting the wrong data base, a user will be asked to confirm the entry on the SAMS:Compress DBD DATA DELETION CONFIRMATION screen. The DBD name, the data base title, and other pertinent information will be presented. The default value for the confirmation field is N, for no delete. Take a moment to verify that this is the correct data base to delete. If it is, enter a Y in the confirmation field and press <ENTER>. The information about the data base will then be erased from the control file.

If you enter the wrong DBD name and need to cancel the delete request, simply press <ENTER> or the <END> PF key.

If you delete information on a data base that has already been compressed under Express, you will not cause any harm.

Changing a Data Base Under the Control of Express

If any changes are to be made to the structure of the IMS data base already compressed with Express, follow the normal IMS procedures that must be performed for the type of data base modifications desired.

Chapter 5. Interactive Dialog Reference

The Interactive Dialog is a menu-driven user facility used to implement SAMS:Compress. This chapter of the User's Guide describes the Interactive Dialog. The first section describes the process whereby a lost or damaged FDT can be recreated through the Dialog. The next section contains illustrations of the Interactive Dialog Panels.

The panels are presented in a sequence following their branching logic; that is, the high-level panel is presented first and is followed by the first second-level panel, which is succeeded by all panels accessed through this second-level panel. Then the next second-level panel is presented, followed by all panels accessed through its options. This sequence is continued until all levels of panels are presented.

This order does not necessarily follow the order in which a user would select the panels when implementing SAMS:Compress. For step-by-step procedures, refer to the following chapters as needed: "Implementation Considerations," "Implementing Express," or "Installation." This chapter is provided for reference should questions arise while a user is performing the procedures.

Recreating a Lost or Damaged FDT

FDTs are generated into the Interactive Dialog FDTLIB. The information used to create FDTs and a copy of the FDT are stored in the Interactive Dialog Control File.

As each FDT is created, it is assigned a control number that is created and maintained by the Dialog. The control number, more commonly referred to as the Integrity Check Block (ICB), identifies a specific version of the FDT, no matter how many times that FDT is created. Each FDT has a unique ICB which becomes part of each segment or record compressed. The ICB contains a 14-bit binary number; given the length of this binary number, the Dialog can generate and identify 16,383 unique FDTs.

The Production FDT Services of the Interactive Dialog provide the user with the ability to list all FDTs, and to regenerate any FDTs for which the original load module has been destroyed.

To regenerate an FDT, perform the following steps:

1. Select option 8 from the Primary Services Menu of the Interactive Dialog.
2. Select option I to Generate a Stored FDT Index List.
3. Determine the FDT ID of the desired FDT by finding the name of the FDT on the list. Since several versions of an FDT may exist, verify the following:
 - the DBD name of the file code
 - the date of creation
 - the name of the creator
4. Select option 1 from the Primary Services Menu.
5. Enter the FDT ID.
6. The information regarding the FDT will be displayed. You must confirm that the FDT is correct. Enter a Y or an N as appropriate.
7. The FDT is now generated and placed in the Interactive Dialog FDTLIB. Move the new FDT to the library where your production FDTs reside.

SAMS:Compress Interactive Dialog Panel - SFEP\$000

```

SAMS:Compress for IMS -- INTERACTIVE DIALOG -- LOGO PANEL -----
COMMAND ==>

-

          SAMS:                                DATE ==> 94/01/10
        CCCC                                TIME ==> 15:58
      CCCCCC                                0000
    CCCCCCCC                                000000    M      M
  CCC   CCC  OO   OO  M      M  P P P P P P
CCC   CC   OO   OO  MM   MM  P   P   R R R R R
CCC   OO   OO   OO  MMM  MMM  P   P   R   R  E E E E E
CCC   OO   OO   OO  M  MM  MM  M  P   P   R   R  E   S S S
CCC   OO   OO   OO  M  M  M  M  P   P   R   R  E   S S  S S
CCC   OO   OO   OO  M      M  M  P   P   R   R  E   S S  S
CCC   OO   OO   OO  M      M  P P P P P P  R R R R R  E E E E  S S  S
CCC   OO   OO   OO  M      M  P   RR   E E E E  SS   S S
CCC   OO   OO   OO  M      M  P   R R   E   SS  S S
CCC   OO   OO   OO  M      M  P   R R   E   S S  S S
CCC   OO   OO   OO  M      M  P   R R   E   S S S
CCC   OO   OO   OO  M      M  P   R   R  E E E E E
CCC   CC   OO   OO  M      M  P   R   R
CCC   CCC  OO   OO  M      M  P                                for IMS
CCCCCCCCC  000000  M      M                                RELEASE: 5.2.3
  CCCCCC  0000
  CCCC
          COPYRIGHT(C) 1994 STERLING SOFTWARE, INC.
  
```

This panel is displayed during the initialization and termination of the SAMS:Compress Interactive Dialog.

Unknown User ID Panel - SFEP\$UUI

```
SAMS:Compress for IMS -- INTERACTIVE DIALOG -- UNKNOWN USER ID -----  
COMMAND ===  
  
-  
  
DATE === 94/01/10  
TIME === 16:02  
  
BECAUSE OF INTERNAL TRACKING, ALL USERS OF THE SAMS:Compress  
INTERACTIVE DIALOG MUST BE DEFINED TO THE SYSTEM AND TSO  
USERID _____ IS NOT CURRENTLY DEFINED.  
  
IF YOU FEEL THAT YOU NEED TO USE THIS FACILITY TO DEFINE AND/OR  
MAINTAIN SAMS:Compress CONTROL DATA, PLEASE CONTACT THE FOLLOWING  
PERSON AND EXPLAIN YOUR POSITION AND NEEDS:  
  
THE USER  
MUST ENTER  
THE NAME OF  
THE PERSON TO  
CONTACT IN THIS SPACE.  
  
PRESS THE 'ENTER' KEY OR THE 'END' PFKEY TO RETURN TO THE  
SAMS:Compress INTERACTIVE DIALOG MANAGER AND TERMINATE THE CURRENT  
DIALOG SESSION.
```

This panel is displayed when someone who has not been defined as an authorized user of the SAMS:Compress Interactive Dialog attempts to use the Dialog.

Description of Field

USERID—The TSO user id of the person currently logged on to TSO.

Report Printing Option Panel - SFEPHCPY

```

SAMS:Compress for IMS -- INTERACTIVE DIALOG -- REPORT PRINTING OPTION -----
COMMAND ==>>

                                     DATE ==>> 94/01/10
                                     TIME ==>> 16:04

HARDCOPY OPTION ==>> N
-

N -- DO NOT PRINT THE REPORT.
Y -- PRINT THE REPORT TO THE SPECIFIED SYSOUT CLASS FOR THE
    SPECIFIED NUMBER OF COPIES AT THE SPECIFIED DESTINATION.

SYSOUT CLASS ==>>> A          ANY VALID SYSOUT CLASS

NUMBER OF COPIES ==>> 1          CAN BE 1 THRU 9 COPIES
DESTINATION ==>>> LOCAL___    LOCAL/REMOTE ROUTING NODE
USER ID ==>>>>>>> ISPMML_    ROUTE TO USER ID

PRESSING THE 'END' PFKEY IS THE SAME AS ENTERING AN 'N'.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is displayed when a user terminates the BROWSE of a report generated by SAMS:Compress Interactive Dialog. A user can obtain a print-out of the report. If a report has separate browse and print options, it is only presented for the print option.

Description of Fields

HARDCOPY OPTION—Points the report to a hard copy printer or not.

SYSOUT CLASS—The class to which the report is to be routed.

NUMBER OF COPIES—Number of copies of the report to be printed. This field will be initialized to 1 but can be change to any single-digit number except 0.

DESTINATION—The destination to which the report is to be routed. This field is initialized at the beginning of the dialog session to LOCAL, which causes the report to be printed on the system printer attached to the CPU. If you want the report printed on a remote printer, you must enter the destination code for the desired printer. This field corresponds to the DEST parameter in a SYSOUT DD statement. This parameter initially defaults to LOCAL, but once it is changed, it will be retained from session to session.

USER ID—This field is initialized to the current user's TSO ID at the beginning of the dialog session. If the destination field is LOCAL or REMOTE, this field is ignored. It is only used for distributed printing. This field should contain the identity code of the person to receive the report at the remote destination.

Control File I/O Error Panel - SFEPFCFIO

```
SAMS:Compress for IMS -- DIALOG -- CONTROL FILE I/O ERROR -----  
COMMAND ==>>  
  
AN ERROR OCCURRED WHILE TRYING TO PEFORM AN I/O OPERATION WITH  
THE SAMS:Compress INTERACTIVE DIALOG CONTROL FILE. A SNAPDUMP OF  
CRITICAL STORAGE AREAS WILL WILL BE SENT TO SYSOUT CLASS 'A'.  
  
INSTALLATION:  
USER:                                DATE:  94/01/24  TIME:  15:06  
REQUESTING PGM ID:                    NAME:  
  
DDNAME:  SFECNTL  DSNAME:  ISPTKA1.DIAL52.CONTROL.FILE  
OPERATION REQUESTED:  
  
RPL RETURN CODE:      COMPONENT CODE:      REASON CODE:      ERROR CODE:  
RPL OPTCD:  
ACB ERROR:  
  
TO PRINT THIS DISPLAY ON YOUR ISPF LIST DATA SET, TYPE 'PRINT' AND  
PRESS THE 'ENTER' KEY.  
  
PRESS THE 'ENTER' KEY TO CONTINUE  
  
- PRESS THE 'HELP' PFKEY FOR ADDITIONAL INFORMATION -----
```

This panel is displayed if an I/O error is encountered while the Dialog was accessing the SAMS:Compress Interactive Dialog Control File.

Description of Fields

SYSOUT CLASS—The SYSOUT class to which the SNAP-DUMP was routed.

INSTALLATION—The name of the installation which encountered the error.

USER—The name of the user at the time the error occurred.

DATE—The date the error occurred.

TIME—The time the error occurred.

REQUESTING PGM ID—The ID of the program/module in which the error occurred.

NAME—The title/name of the failing program/module.

DDNAME—The ddname of the Dialog Control File.

DSNAME—The data set name of the installation's Dialog Control File.

OPERATION REQUESTED—The type of operation that was being attempted.

RPL RETURN CODE—The VSAM return code from the RPL.

COMPONENT CODE—The VSAM component code from the RPL.

REASON CODE—The VSAM reason code from the RPL.

ERROR CODE—The VSAM error code from the RPL.

RPL OPTCD—The operation code from the RPL.

ACB ERROR—The VSAM error code from the ACB.

Data Set Allocation Error Panel - SFEPALER

```
SAMS:Compress for IMS -- DIALOG -- DATA SET ALLOCATION ERROR -----  
COMMAND ===>  
  
-  
  
AN ERROR OCCURRED WHILE TRYING TO ALLOCATE A DATA SET.  PRESS  
THE 'ENTER' KEY TO HAVE THIS PANEL PRINTED TO YOUR ISPF LIST DATA SET  
AND TO CONTINUE PROCESSING.  A SNAPDUMP OF CRITICAL STORAGE AREAS  
WILL BE SENT TO SYSOUT CLASS 'A'.  
  
INSTALLATION:  
USER:                                DATE:  94/01/10  TIME:  16:15  
REQUESTING PGM ID:                    NAME:  
  
DDNAME:                                UNIT:  
DSNAME:  
ALLOCATION RETURN CODE:                ERROR CODE:                INFO CODE:  
  
PRESS THE 'HELP' PFKEY TO REVIEW THE COMMON ALLOCATION ERRORS.  
  
TO PRINT THIS DISPLAY ON YOUR ISPF LIST DATA SET, TYPE 'PRINT' AND  
PRESS THE 'ENTER' KEY.  
  
PRESS THE 'ENTER' KEY TO CONTINUE  
  
- PRESS THE 'HELP' PFKEY FOR ADDITIONAL INFORMATION -----
```

The purpose of this panel is to display a data set allocation error to the user.

Description of Fields

SYSOUT CLASS—The SYSOUT class to which the SNAP-DUMP was routed.

INSTALLATION—The name of the installation which encountered the error.

USER—The name of the user at the time the error occurred.

DATE—The date the error occurred.

TIME—The time the error occurred.

REQUESTING PGM ID—The ID of the program/module in which the error occurred.

NAME—The title/name of the failing program/module.

DDNAME—The DDNAME to which the file was to be allocated.

UNIT—The type of unit on which the data set resides.

DSNAME—The name of the data set to be allocated.

ALLOCATION RETURN CODE—The SVC 99 return code.

ERROR CODE—The SVC 99 error code.

INFO CODE—The SVC 99 error code information.

Primary Services Menu - SFEPMENU

```
SAMS:Compress for IMS -- INTERACTIVE DIALOG -- PRIMARY SERVICES MENU -----  
COMMAND ===>  
  
-  
  
OPTION ===>                                     DATE ===> 94/01/10  
                                                TIME ===> 16:12  
1 -- SAMS:Compress for IMS SUPPORT (FULL FUNCTION, FASTPATH AND EXPRESS)  
  
8 -- PRODUCTION PDT SERVICES.  
9 -- MAINTAIN SAMS:Compress INTERACTIVE DIALOG CONTROL DATA.  
X -- EXIT SAMS:Compress INTERACTIVE DIALOG.  
  
ENTER THE CODE FOR THE DESIRED SAMS:Compress SUPPORT SERVICE AND  
PRESS THE 'ENTER' KEY.  
  
PRESSING THE 'END' PFKEY IS THE SAME AS SELECTING THE 'X' OPTION.  
  
----- PRESS THE 'HELP' PFKEY TO ENTER THE DIALOG TUTORIAL -----
```

This is the primary menu for the SAMS:Compress Interactive Dialog.

Description of Field

OPTION—This field is used to indicate which of the service areas is to be invoked.

FDT Support Services Menu - SFEP8MNU

```

SAMS:Compress for IMS -- FDT SUPPORT SERVICES MENU -----
COMMAND ==>

OPTION ==>                                DATE ==> 94/01/10
      -                                     TIME ==> 16:15

1 -- REGENERATE AN FDT.

4 -- DELETE AN OBSOLETE FDT FROM THE DIALOG CONTROL FILE.

I -- GENERATE A STORED FDT INDEX LIST.

X -- RETURN TO THE SAMS:Compress INTERACTIVE DIALOG PRIMARY MENU.

ENTER THE CODE FOR THE DESIRED FDT SERVICE AND PRESS THE 'ENTER' KEY.

PRESSING THE 'END' PFKEY IS THE SAME AS SELECTING THE 'X' OPTION.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is displayed when the FDT services are selected on the primary menu or when one of the FDT services has completed its processing.

Description of Field

OPTION—This field allows the user to select the desired FDT service.

FDT Regeneration Confirmation Panel - SFEP8FGC

```
SAMS:Compress for IMS -- FDT REGENERATION CONFIRMATION -----  
COMMAND ==>>  
  
DATE ==>> 94/01/10  
TIME ==>> 16:17  
  
FDT ID: FDT00053   FDT NAME: SEGMENTF  
  
FDT FOR: SAMPLEDB -- TEST  
          SEGMENTF  
  
CREATED BY: MARY MARIE MELLMANN      DATE: 01/06/94   TIME: 12:55  
  
CONFIRM REGENERATION ==>> N      'N' -- DO NOT REGENERATE THE FDT  
                               -  
                               'Y' -- REGENERATE THE FDT  
  
IF THIS IS NOT THE FDT TO BE REGENERATED, PRESS THE 'ENTER' KEY.  
  
IF THIS IS THE FDT TO BE REGENERATED, ENTER A 'Y' IN THE CONFIRM  
REGENERATION FIELD AND PRESS THE 'ENTER' KEY.  
  
PRESSING THE 'END' PFKEY IS THE SAME AS PRESSING THE 'ENTER' KEY  
WITHOUT CHANGING THE CONFIRM REGENERATION FIELD.  
  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is used to provide the user with information about the specified FDT so that the user can confirm that the correct FDT will be re-generated.

Description of Fields

FDT ID—The identity code for the FDT.

FDT NAME—The load module name of the FDT.

FDT FOR—The DBD for which the FDT was created, the title of the DBD or File for which the FDT was originally created, and the segment name for which the FDT was created.

CREATED BY—The name of the person who originally created the FDT.

DATE—The date the FDT was originally created.

TIME—The time the FDT was originally created.

CONFIRM REGENERATION—FDT re-generation confirmation code field.

FDT Deletion Confirmation Panel - SFEP8DDC

```

SAMS:Compress for IMS -- FDT DELETION CONFIRMATION -----
COMMAND ==>>

                                     DATE ==>> 94/01/10
                                     TIME ==>> 16:18

FDT ID: FDT00053   FDT NAME: SEGMENTF

FDT FOR: SAMPLEDB -- TEST
          SEGMENTF

CREATED BY: MARY MARIE MELLMANN      DATE: 01/06/94   TIME: 12:55

CONFIRM DELETE ==>> N      'N' -- DO NOT DELETE THE FDT.
                        -
                        'Y' -- DELETE THE FDT.

IF THIS IS NOT THE FDT TO BE DELETED, PRESS THE 'ENTER' KEY.

IF THIS IS THE FDT TO BE DELETED, ENTER A 'Y' IN THE CONFIRM DELETE
FIELD AND PRESS THE 'ENTER' KEY.

PRESSING THE 'END' PFKEY IS THE SAME AS PRESSING THE ENTER KEY
WITHOUT CHANGING THE CONFIRM DELETE FIELD.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is presented when the user has identified an FDT to be deleted. It provides detailed information about the FDT and requires that the user confirm that the data shown is really for the FDT to be deleted.

Description of Fields

FDT ID—FDT identity code.

FDT NAME—FDT load module name.

FDT FOR—The DBD segment name for which the FDT was generated, and the title of the data base for which the FDT was generated.

CREATED BY—The name of the person that generated the FDT.

DATE—The date the FDT was generated.

TIME—The time the FDT was generated.

CONFIRM DELETE—Deletion confirmation field.

Control Maintenance Menu - SFEP9MNU

```
SAMS:Compress for IMS -- CONTROL MAINTENANCE MENU -----  
COMMAND ===>  
  
OPTION ===>                                     DATE ===> 94/01/10  
-                                                                 TIME ===> 16:18  
  
1 -- MAINTAIN INSTALLATION PARAMETERS.  
2 -- DEFINE/MAINTAIN SAMS:Compress INTERACTIVE DIALOG USER DATA.  
3 -- MAINTAIN DEFAULT DATA SET NAMES.  
L -- LIST THE DEFINED USERS.  
X -- RETURN TO THE SAMS:Compress INTERACTIVE DIALOG PRIMARY MENU.  
  
ENTER THE CODE FOR THE DESIRED SAMS:Compress SUPPORT SERVICE AND  
PRESS THE 'ENTER' KEY.  
  
PRESSING THE 'END' PFKEY IS THE SAME AS SELECTING THE 'X' OPTION.  
  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is of the SAMS:Compress Interactive Dialog control maintenance services menu. It is displayed when option 'C' is selected on the primary menu panel, and after the completion of any of the control maintenance services.

Description of Field

OPTION—Allows the user to indicate which of the control maintenance services is to be invoked.

Maintain Installation Control Parameters Panel - SFEP9CMP

```

SAMS:Compress for IMS -- MAINTAIN INSTALLATION CONTROL PARAMETERS -----
COMMAND ==>>

                                                    DATE ==>> 94/01/10
                                                    TIME ==>> 16:19

INSTALLATION NAME ==>> STERLING SOFTWARE, INC. V.5.2.3_____
-
SECURED SERVICES ==>>> N                Y = YES AND N = NO
DEFAULT DASD UNIT ==>>> SSLDA_____
DEFAULT TAPE UNIT ==>>> TAPE_____
CURRENT DFP LEVEL ==>>> 33

CURRENCY UNIT ==>>>>> DOLLARS_
DASD PURCHASE COST ==>>>> 0000024.00 PER MEGABYTE
DASD CHARGE/MO ==>>>>> 0000001.25 PER MEGABYTE
PROCESSOR SPEED ==>>>>> 016          IN MILLIONS OF INSTRUCTIONS/SECOND

CPU/EXCP CHARGES:      BATCH          ONLINE          EXCP
-----
PRIME ==>>>>>>>>>>>>> 0000890.00 -- 0002200.00 -- 0000000.50
OVERNITE ==>>>>>>>>>>>>> 0000089.00 -- 0000220.00 -- 0000000.05

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

Displayed when Maintain Installation Parameter service is selected on the Control Maintenance menu, this panel allows users to record standard values used for report and JCL generation, and for calculation of DASD utilization and cost savings.

Description of Fields

INSTALLATION NAME—Company/installation name.

SECURED SERVICES—Security option.

DEFAULT DASD UNIT—Default DASD logical unit type.

DEFAULT TAPE UNIT—Default TAPE logical unit type.

CURRENT DFP LEVEL—Current level of Data Facility Product (DFP) installed.

CURRENCY UNIT—Currency/monetary unit used by the company.

DASD PURCHASE COST—Cost of procuring new DASD expressed in cost per megabyte of storage.

DASD CHARGE/MO—Amount users are charged for the use of one megabyte of DASD per month.

PROCESSOR SPEED—The average speed of the CPU processors in the computers at the installation. The speed is expressed in Millions of Instructions Per Second (MIPS). Calculate the total processing MIPS at the installation and divide it by the number of processors. The processor speed should be rounded to the nearest MIPS.

CPU/EXCP CHARGES—The remaining items are under this heading.

PRIME (BATCH)—Amount charged for the use of one hour of CPU time during prime operating hours for BATCH processing.

PRIME (ONLINE)—Amount charged for the use of one hour of CPU time during prime operating hours for ON-LINE processing.

PRIME (EXCP)—Amount charged for 1,000 EXCPs during prime operating hours.

OVERNIGHT (BATCH)—Amount charged for the use of one hour of CPU time during non-prime operating hours for BATCH processing.

OVERNIGHT (ONLINE)—Amount charged for the use of one hour of CPU time during non-prime operating hours for ON-LINE processing.

OVERNIGHT (EXCP)—Amount charged for 1,000 EXCPs during non-prime operating hours.

User Maintenance Menu - SFEP9UID

```

SAMS:Compress for IMS -- USER MAINTENANCE MENU -----
COMMAND ==>>

      TSO ID =====>> _____                DATE ==>> 94/01/10
      -
                                           TIME ==>> 16:19

TYPE OF MAINT ==>> C
  A -- ADD A NEW USER.
  C -- CHANGE USER DATA.
  D -- DELETE A USER.

      X -- RETURN TO THE CONTROL MAINTENANCE MENU.

PRESS THE 'END' PFKEY TO RETURN TO THE CONTROL MAINTENANCE MENU.
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

Displayed when define/maintain SAMS:Compress Interactive Dialog User data option on the Control Maintenance Menu panel is selected, and after completion of user data maintenance, this panel controls user data maintenance functions.

Description of Fields

TSO ID—The TSO ID of the authorized user. Required for options A, C, and D.

TYPE OF MAINT—The code for the type of user maintenance desired. The Change option is displayed as the the initial option.

User Data Maintenance Panel - SFEP9USR

```
SAMS:Compress for IMS -- USER DATA MAINTENANCE -----  
COMMAND ==>  
  
TSO ID =====> ISPMMM2                                DATE ==> 94/01/10  
                                                    TIME ==> 16:20  
  
FULL NAME =====> MARY MARIE MELLMANN_____
```

—

```
MESSAGE NAME ==> MMM_____
```

```
MAY USE FDT SUPPORT =====> Y
```

```
MAY USE CONTROL MAINT ==> Y
```

ENTER THE USER'S TSO ID, NAME (LAST, FIRST, MI.), HOW THE USER SHOULD
BE ADDRESSED IN ANY MESSAGES OR PANELS, AND INDICATE WHICH OF THE
FACILITIES HE/SHE MAY USE. FACILITY USAGE WILL BE INDICATED BY
AN 'N' FOR NO OR A 'Y' FOR YES. AFTER ALL DATA IS ENTERED, PRESS
THE 'ENTER' KEY TO RECORD THE USER DATA.

PRESS THE 'END' PFKEY TO RETURN TO THE USER MAINTENANCE MENU.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

This panel is displayed when the add or change option is selected from the User Parameter Maintenance Menu. When the Add option is selected, only the TSO id will contain data. When the Change option is selected, all fields will contain current data.

Description of Fields

TSO ID—The TSO ID of the user.

FULL NAME—The full name of the user (last, first, mi) is suggested.

MESSAGE NAME—The first name or nickname of the user.

MAY USE FDT SUPPORT—Authorized to use the FDT services (Yes or No).

MAY USE CONTROL MAINT—Authorized to perform Control Maintenance (Yes or No).

User Deletion Confirmation Panel - SFEP9UDC

```

SAMS:Compress for IMS -- USER DELETION CONFIRMATION -----
COMMAND ==>>

      TSO ID =====> ISPMMM2                                DATE ==>> 94/01/10
                                                                TIME ==>> 16:23
      FULL NAME =====> MARY MARIE MELLMANN

      MESSAGE NAME ==>> MMM

      DELETE CONFIRM CODE ==>> N
                               -
      N -- NO, THIS IS NOT THE PERSON TO BE DELETED.
      Y -- YES, THIS IS THE PERSON TO BE DELETED.

      ENTER A Y OR AN N IN THE DELETE CONFIRMATION CODE AND PRESS THE 'ENTER' KEY.

      PRESSING THE 'END' PFKEY WILL HAVE THE SAME EFFECT AS ENTERING AN
      'N' IN THE DELETE CONFIRMATION CODE AND PRESSING THE 'ENTER' KEY.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is displayed for the maintenance request to delete a user. All fields contain the current data for the user. The delete confirmation field is initialized to 'N'.

Description of Fields

TSO ID—User's TSO ID.

FULL NAME—User's full name (last, first, MI).

MESSAGE NAME—User's message name.

DELETE CONFIRM CODE—Deletion confirmation indicator.

Define Default Data Sets Panel - SFEP9DSN

```
SAMS:Compress for IMS -- DEFINE DEFAULT DATA SETS -----  
COMMAND ==>>>  
  
ENTER OR MODIFY THE DEFAULT DATA SET NAMES AND THEN      DATE ==>> 94/01/10  
PRESS THE 'ENTER' KEY.                                     TIME ==>> 16:23  
  
FDT LIBRARY ==>>>> ISPMMML.FDTLIB_____   
-   
ACB LIBRARY ==>>>> SHRINK.ACBLIB_____   
DBD LIBRARY ==>>>> SHRINK.IMS.DBDLIB_____   
PSB LIBRARY ==>>>> SHRINK.IMS.PSBLIB_____   
IMS RESLIB ==>>>> IMSESA.SYSS.RESLIB_____   
DBD SOURCE ==>>>> ISPMMML.GENLIB_____   
IMS IMSID PARM ==>> SYSS                (OPTIONAL)   
IMS AGN PARM ==>>> IMSSPG__            (OPTIONAL)  
  
  
PRESS THE 'END' PFKEY TO CANCEL THE UPDATING OF THE DEFAULT DATA SET NAMES.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed when the Maintain Default Data Set Names option is selected on the Control Maintenance Menu, and allows the user to supply the default information provided to each user at the beginning of every Dialog session.

Description of Fields

FDT LIBRARY—Data set name of default FDT library. A required field.

ACB LIBRARY—Data set name of default ACB library. A required field.

DBD LIBRARY—Data set name of default DBD library. A required field.

PSB LIBRARY—Data set name of default PSB library. A required field.

IMS RESLIB—Data set name of default IMS RESLIB. A required field.

DBD SOURCE—Data set name of DBD source macros library. A required field.

IMS IMSID PARM—The default value for the IMSID parameter in the IMS EXEC parameter list. This entry is optional for the Interactive Dialog but can be required at an installation depending upon how IMS has been installed.

IMS AGN PARM—The default value for the AGN parameter in the IMS EXEC parameter list. This entry is optional for the Interactive Dialog but can be required at an installation depending upon how IMS has been installed.

IMS Support Services Menu - SFEP1MNU

```

SAMS:Compress for IMS -- IMS SUPPORT SERVICES MENU -----
COMMAND ==>>

OPTION ==>>                                     DATE ==>> 94/01/10
      -                                           TIME ==>> 16:24

1 -- ANALYZE A DATA BASE.
2 -- GENERATE A STORED DATA REPORT.
3 -- MODIFY USER PARAMETERS.
4 -- GENERATE JCL FOR TESTING.
5 -- IMPLEMENT DATA BASE UNDER SAMS:Compress for IMS CONTROL.

8 -- DELETE AN OBSOLETE DATA BASE FROM DIALOG CONTROL FILE.

I -- GENERATE A DATA BASE INDEX REPORT.
X -- RETURN TO THE SAMS:Compress INTERACTIVE DIALOG PRIMARY MENU.

ENTER THE CODE FOR THE DESIRED IMS DATA BASE SERVICE AND PRESS
THE 'ENTER' KEY.

PRESSING THE 'END' PFKEY IS THE SAME AS SELECTING THE 'X' OPTION.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
    
```

This is the services menu for SAMS:Compress for IMS.

Description of Field

OPTION—This field is where the code for the desired service is entered.

IMS Data Base Analysis Panel - SFEP1ANA

```
SAMS:Compress for IMS -- IMS DATA BASE ANALYSIS -----  
COMMAND ===>  
  
DATE ==> 94/01/10  
TIME ==> 16:24  
  
DBD NAME ==> _____  
-  
  
DBDLIB ===== SHRINK,IMS,DBDLIB_____  
DATA BASE TITLE ==> _____  
  
ENTER THE DBD NAME OF THE IMS DATA BASE WHICH IS TO BE ANALYZED.  
  
IF A DIFFERENT DBDLIB DATA SET IS TO BE USED FOR THE ANALYSIS,  
ENTER THE CORRECT DBDLIB DATA SET NAME.  
  
PRESS THE 'END' PFKEY TO RETURN TO THE IMS SUPPORT SERVICES MENU.  
  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel allows the user to identify the DBD whose structure is to be analyzed and the data set name of the library where the DBD load module can be found.

Description of Fields

DBD NAME—Name of the DBD.

DBDLIB—Name of the DBDLIB data set where the DBD load module is to be found.

DATA BASE TITLE—The descriptive title of the data base.

IMS DBD Analysis Duplicate DDNAME Confirmation Panel - SFEPDUPD

```

SAMS:Compress for IMS -- IMS DBD ANALYSIS DUPLICATE DDNAME CONFIRMATION -----
COMMAND ==>>

                                                    DATE ==>> 94/01/10
                                                    TIME ==>> 16:24

DBD NAME:
DATA BASE TITLE:

DDNAME:           SOURCE DBD NAME:

CONFIRM DUPLICATE ==>> _      'N' -- DDNAME IS NOT FOR A SHARED INDEX.
                    -
                    'Y' -- DDNAME IS FOR A SHARED INDEX.

THE SHOWN DDNAME HAS ALREADY BEEN RECORDED FOR THE DATA BASE BEING ANALYZED.
IF THE SOURCE DBD NAME IS FOR A SHARED SECONDARY INDEX AND THE DDNAME IS
CORRECT, ENTER A 'Y' IN THE CONFIRM DUPLICATE FIELD AND PRESS THE ENTER KEY.

IF THE SOURCE DBD NAME IS NOT FOR A SHARED SECONDARY INDEX, ENTER
A 'N' IN THE CONFIRM DUPLICATE FIELD AND PRESS THE ENTER KEY.

PRESSING THE END PFKEY IS THE SAME AS PRESSING THE ENTER KEY WITHOUT
CHANGING THE CONFIRM DUPLICATE FIELD.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
    
```

This panel is displayed during DBD analysis if the same DDNAME is found more than once. When this panel is displayed, the user must examine the data and indicate whether the DDNAME in question belongs to a shared secondary index or not.

Description of Fields

DBD NAME—DBD name of the data base being analyzed.

DATA BASE TITLE—Title of the data base being analyzed.

DDNAME—DDNAME that has been found more than once.

SOURCE DBD NAME—The name of the DBD in which the multiple occurrence was discovered.

CONFIRM DUPLICATE—This field is where the user indicates whether the DDNAME is part of a shared secondary index or not.

IMS Database Stored Data Report Generation Panel - SFEP1RPT

```
SAMS:Compress for IMS -- IMS DATABASE STORED DATA REPORT GENERATION -----  
COMMAND ===>  
  
DATE ===> 94/01/10  
TIME ===> 16:25  
  
DBD NAME ===> _____  
-  
  
ENTER THE DBD NAME OF THE IMS DATA BASE WHICH IS TO BE REPORTED.  
PRESS THE "END" PFKEY TO RETURN TO THE IMS SERVICE MENU.  
  
----- PRESS THE "HELP" PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed when the user has selected the Stored Data Report option on the IMS SERVICES menu panel. It is displayed so that the user can specify the DBD for which the report is to be generated.

Description of Field

DBD NAME—The DBD name of the data base whose data is to be reported.

IMS User Parameter Maintenance Menu - SFEP1UPM

```

SAMS:Compress for IMS -- IMS USER PARAMETER MAINTENANCE MENU -----
COMMAND ===>

                                     DATE ===> 94/01/10
                                     TIME ===> 16:26

DBD NAME ===> _____
-

TYPE OF MAINTENANCE ===> _
1 -- DATA BASE TITLE.
2 -- DATA SAMPLE EXTRACT PARAMETERS.
3 -- RDL USER PARAMETERS.
4 -- FDT USER PARAMETERS.
5 -- DATA SET NAME(S).
6 -- SEGMENT COUNTS (DO NOT USE UNTIL AFTER DATA SAMPLE IS COMPLETE).

X -- TERMINATE USER PARAMETER MAINTENANCE.

ENTER THE NAME OF THE DBD, ENTER THE TYPE OF MAINTENANCE DESIRED,
AND THEN PRESS THE 'ENTER' KEY.

PRESS THE 'END' PFKEY TO RETURN TO THE IMS SERVICE MENU.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is displayed when the user selects the User Parameter Maintenance option of the IMS SERVICES menu. This sub-menu allows the user to select the type of parameter maintenance to be performed.

Description of Fields

DBD NAME—DBD of the data base whose parameters are to be modified.

TYPE OF MAINTENANCE—The code for the selected maintenance service.

IMS Database Title Maintenance Panel - SFEP1UDT

```
SAMS:Compress for IMS -- IMS DATABASE TITLE MAINTENANCE -----  
COMMAND ==>>  
  
DATE ==> 94/01/10  
TIME ==> 16:27  
  
DBD NAME ==>>  
  
DBD TITLE ==> _____  
                -  
  
ENTER THE CORRECT DATA BASE TITLE AND PRESS THE ENTER KEY.  
PRESS THE 'END' PFKEY TO CANCEL THE DATA BASE TITLE MAINTENANCE.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel allows the user to modify the title of the data base.

Description of Fields

DBD NAME—DBD name of the data base.

DBD TITLE—Title of the data base.

IMS Data Sample Extract Parameter Maintenance Panel - SFEP1UEP

```

SAMS:Compress for IMS -- IMS DATA SAMPLE EXTRACT PARAMETER MAINTENANCE -----
COMMAND ==>>

ENTER THE DESIRED EXTRACTION PARAMETERS AND PRESS THE      DATE ==>> 94/01/10
ENTER KEY TO RECORD THEM. ONLY THE ROOT SEGMENT WILL      TIME ==>> 16:28
BE DISPLAYED. FOR EACH ROOT SEGMENT EXTRACTED, ALL OF
ITS DEPENDENT SEGMENTS WILL ALSO BE EXTRACTED.

DBD:                --
SFEPIMSE PARAMETERS:
SEG NAME  BYPASS  EXTRACT  SKIP
-----  -
          -
          -
          -

----- PRESS THE HELP PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
    
```

This panel allows the user to establish data sample extract parameters other than those established during DBD analysis.

Description of Fields

DBD—DBD name and title of the data base.

SFEPIMSE PARAMETERS—The remaining items are under this heading.

SEG NAME—Name of the data base’s root segment.

BYPASS—The number of data base records to be ‘BYPASSED’ before the extraction of the data sample begins. This number is expressed in multiples of 1,000. The field has an initial default of 0.

EXTRACT—Indicates how many data base records are to be extracted to form the data sample. The number is expressed as a multiple of 1,000. The default value of ‘ALL’ means that once the sample extraction begins, it is to progress ‘ALL’ the way through the data base.

SKIP—This field is used to indicate the number of data base records to be SKIPPed between each data base record that is extracted.

RDL Maintenance Selection List Panel - SFEP1URL

```
SAMS:Compress for IMS -- IMS USER RDL PARAMETER MAINTENANCE -----
COMMAND ==>>                                SCROLL ==>> CSR

INDICATE THE SEGMENT(S) WHOSE RDL YOU WISH TO WORK          DATE ==>> 94/01/10
ON BY PLACING ANY CHARACTER IN THE SELECT FIELD            TIME ==>> 16:29
IN FRONT OF THE SEGMENT NAME AND THEN PRESSING THE 'ENTER' KEY.
USE THE FORWARD AND BACKWARD PFKEYS TO SCROLL THROUGH THE SEGMENT TABLE.
PRESS THE 'END' PFKEY TO TERMINATE THE RDL MAINTENANCE.

DBD:                --
-- SEG NAME:         KEY NAME:         DATA:         COMPRESS BYTES:
--
```

This panel provides users with a list of segments in the data base from which to select one segment at a time to modify the RDL of that segment.

Description of Fields

DBD—The DBD name and title of the data base.

SEG NAME—Name of the data base segment.

KEY NAME—Name of the key field of the segment, if any.

DATA—The start position and length of the segment key field.

COMPRESS BYTES—The number of compressible bytes of data found in the segment.

IMS User RDL Parameter Maintenance Panel - SFEP1URP

```

SAMS:Compress for IMS -- IMS USER RDL PARAMETER MAINTENANCE -----
COMMAND ==>>

MODIFY THE NONKEY PORTION OF THE RECORD DEFINITION      DATE ==>> 94/01/10
OF THE SEGMENT.  PRESS THE ENTER KEY TO EDIT THE RDL.   TIME ==>> 16:29

DBD:              --
SEGMENT: NAME()  FORMAT() MIN LENG() MAX LENG()

THE RECORD DEFINITION CAN BE CHANGED ONLY BEYOND THE KEY FIELD.

-----
-
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

REMEMBER, THE RECORD DEFINITION MUST END WITH A PERIOD '.'.
    
```

This panel displays the current RDL of a selected segment. The user can modify the RDL in fields 8 through 18.

Description of Fields

DBD—DBD name and title of the data base.

SEGMENT—The next five fields are under this heading.

NAME—Segment name.

FORMAT—Segment format (Fixed or Variable).

MIN LENG—Minimum length of the segment (0 if fixed).

MAX LENG—Maximum length of the segment.

Variable length control and segment key data (not modifiable).

Eleven lines for coding the RDL for the compressible portion of the segment.

IMS User FDT Parameter Maintenance Panel - SFEP1UFP

```

SAMS:Compress for IMS -- IMS USER FDT PARAMETER MAINTENANCE -----
COMMAND ==>                                SCROLL ==> CSR

ENTER OR CORRECT THE FDT DATA.  USE THE FORWARD AND BACKWARD PFKEYS
TO SCROLL THROUGH THE SEGMENT TABLE.  THE DIALOG ALLOWS YOU TO
PERFORM UPDATES TO MORE THAN ONE SEGMENT, BUT YOU MUST PRESS ENTER
AFTER EACH SEGMENT YOU MODIFY.

PRESS THE END PFKEY TO RECORD CHANGES AND END MAINTENANCE.

DBD:      --
SEG NAME  FDT NAME  SHR  MIN SIZE  STAT  PAD LEN  PAD TYP
-----
                SRCE DBD  SRCE SEG
                -----
                _____
                -
                _____
    
```

This panel allows users to: modify the FDT name; establish FDT sharing, minimum compressed segment size, segment padding type and quantity; and indicate whether compression statistics are to be produced during actual processing of the data base.

Description of Fields

DBD—DBD name and title of the data base.

SEG NAME—Segment name.

FDT NAME—FDT name.

SHR—SHARE option.

MIN SIZE—Minimum compressed size.

STAT—Compression statistics generation code.

PAD LEN—Number of padded bytes or padding percent.

PAD TYP—Padding type code. 'N' = number, 'P' = percent

SRCE DBD—DBD name of data base which is to be the source of the shared FDT.

SRCE SEG—The name of the segment to be the source of the shared FDT.

IMS User Data Set Name Maintenance Panel - SFEP1UDS

```
SAMS:Compress for IMS -- IMS USER DATA SET NAME MAINTENANCE -----
COMMAND ==>>                                SCROLL ==>> CSR
```

```
ENTER OR CORRECT THE DATA SET NAMES.                DATE ==>> 94/01/10
MULTIPLE DATA SETS CAN BE ENTERED OR CORRECTED AT ONCE.  TIME ==>> 16:30
USE THE FORWARD AND BACKWARD PFKEYS TO SCROLL THROUGH THE DDNAME TABLE.
PRESS THE 'ENTER' KEY TO RECORD THE CHANGES IN THE WORK TABLE.
PRESS THE 'END' PFKEY TO VALIDATE DATA SETS AND RETURN TO MAINTENANCE MENU.
```

```
DBD:                --
```

DBD NAME	DDNAME	DATA SET NAME
-----	-----	-----

		-

This panel is displayed when Data Set Names option on the User Parameter Maintenance sub-menu is selected. The user must enter the actual data set name for each of the DDNAMEs. The data set names are stored in an ISPF table each time the user presses the 'ENTER' key or the 'END' PFkey. When the user presses the 'END' PFkey, the data set names are verified against the system's CATALOG. If the data set name is valid, information about the data set is collected and stored in the Interactive Dialog Control File. If the data set name is not found, the table is redisplayed with an error message. When the user presses the 'HELP' PFkey, the data set name in error is displayed.

Description of Fields

DBD—DBD name and title of the data base being processed.

DBD NAME—DBD name of the data base in which the DDNAME was found.

DDNAME—DDNAME for the data base component data set.

DATA SET NAME—Data set name of the data base component.

IMS Database Segment Count Maintenance Panel - SFEP1USC

```
SAMS:Compress for IMS -- IMS DATABASE SEGMENT COUNT MAINTENANCE -----
COMMAND ==>>

                                         DATE ==>> 94/01/10
                                         TIME ==>> 16:30

DBD:          --

SEG NAME      OCCURRENCES
-----      -----
              _____
              -

ENTER THE TOTAL NUMBER OF RECORDS IN THE DATA BASE, WHICH IS THE SAME
AS THE NUMBER OF ROOT SEGMENTS IN THE DATA BASE.

PRESS THE 'END' PFKEY TO CANCEL THE SEGMENT COUNT MAINTENANCE.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

To be able to project the DASD usage of the total data base from the data sample, the test compression program must know how many occurrences of each segment type there are. This panel is the means by which the user can record the total segment counts.

Description of Fields

DBD—DBD name and title of the data base being processed.

SEG NAME—Name of the data base root segment.

OCCURRENCES—The number of occurrences of a specific segment.

IMS JCL Generation Menu - SFEP1JCL

```
SAMS:Compress for IMS -- IMS JCL GENERATION MENU -----  
COMMAND ==>>  
  
DATE ==>> 94/01/10  
TIME ==>> 16:30  
  
DBD NAME ==>> _____  
-  
  
ENTER THE DBD NAME OF THE DATA BASE FOR WHICH YOU WISH TO GENERATE JCL.  
PRESS THE ENTER KEY TO PROCEED.  
  
PRESS THE END PFKEY TO RETURN TO THE IMS SUPPORT SERVICES MENU.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed when the user elects to generate test JCL. It provides the means for identifying the data base for which the JCL is to be generated.

Description of Field

DBD NAME—DBD name of the data base whose testing JCL is to be generated.

IMS JCL Generation Menu (Full Function) - SFEP1JFF

```
SAMS:Compress for IMS -- IMS JCL GENERATION MENU (FULL FUNCTION) -----  
COMMAND ==>>  
  
DBD NAME: DATE ==>> 94/01/10  
DBD TITLE: TIME ==>> 16:30  
ENTER AND/OR CORRECT THE JOB CARDS.  
JOB CARDS:  
_____  
_____  
_____  
_____  
_____  
_____  
  
MBR NAME PREFIX ==>> _____  
JCL TYPE =====>> _  
  1 -- EXTRACT DATA SAMPLE FROM DATA BASE USING BATCH DLI CALLS.  
  2 -- EXTRACT DATA SAMPLE FROM DATA BASE USING BMP DLI CALLS.  
  3 -- EXTRACT DATA SAMPLE FROM DATA BASE USING BATCH HSSR CALLS.  
  X -- EXIT JCL GENERATION WITHOUT GENERATING ANY JCL.  
JCL WILL BE STORED IN DATASET  
MEMBER NAME WITH 'X' SUFFIX WILL EXTRACT DATA SAMPLE AND MUST BE RUN FIRST.  
MEMBER NAME WITH 'T' SUFFIX WILL PERFORM BYTE DISTRIBUTION AND TEST COMPRESS.  
PRESS THE 'ENTER' TO KEY TO PROCEED.  
PRESS THE 'END' PFKEY TO RETURN TO THE DATA BASE SPECIFICATION SCREEN.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

Displayed if the data base has a Full Function organization (HISAM, HIDAM or HDAM), this panel permits the user to specify the JOB statement to be used with test jobs. Also, it allows the user to specify the first part of the member names that will be generated to contain the data extraction job and the test compression job. Finally, this panel enables the user to select the means by which the data sample will be extracted.

Description of Fields

DBD NAME—DBD name of the data base being processed.

DBD TITLE—Title of the data base being processed.

JOB CARDS—The JOB statement to be used with the test jobs, four lines maximum.

MBR NAME PREFIX—The prefix to be used when creating the JCL member in the Dialogue's GENLIB.

JCL TYPE—Allows the user to specify how the data sample is to be extracted.

JCL WILL BE STORED IN DATA SET—The data set name into which the generated JCL will be stored.

IMS JCL Generation Menu (FASTPATH) - SFEP1JFP

```

SAMS:Compress for IMS -- IMS JCL GENERATION MENU (FASTPATH) -----
COMMAND ==>>

DBD NAME:
DBD TITLE:
ENTER AND/OR CORRECT THE JOB CARDS.
JOB CARDS:
_____
_____
_____
_____

MBR NAME ==>> _____
JCL TYPE ==>> _
  1 -- EXTRACT DATA SAMPLE WITH BMP JOB USING DLI CALLS.
  X -- EXIT JCL GENERATION WITHOUT GENERATING ANY JCL.
JCL WILL BE STORED IN DATASET
MEMBER NAME WITH 'X' SUFFIX WILL EXTRACT DATA SAMPLE AND MUST BE RUN FIRST.
MEMBER NAME WITH 'T' SUFFIX WILL PERFORM BYTE DISTRIBUTION AND TEST COMPRESS.
PRESS THE 'ENTER' KEY TO PROCEED.
PRESS THE 'END' PFKEY TO RETURN TO THE DATA BASE SPECIFICATION SCREEN.
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel is displayed if the data base has a FASTPATH organization. It allows the user to specify the JOB statement to be used with the test jobs and to specify the first part of the member names to be generated to contain data extraction and test compression jobs. Finally, this panel permits the user to select the means by which the data sample will be extracted.

Description of Fields

DBD—DBD name of the data base being processed.

DBD TITLE—Title of the data base being processed.

JOB CARDS—The JOB statement to be used with the test jobs, four lines maximum.

MBR NAME—The prefix to be used when creating the JCL member in the Dialogue's GENLIB.

JCL TYPE—Allows the user to specify how the data sample is to be extracted.

JCL WILL BE STORED IN DATA SET—The data set name into which the generated JCL will be stored.

IMS Data Extraction JCL Generation (Full Function) Panel - SFEP1JD1

```
SAMS:Compress for IMS -- IMS DATA EXTRACTION JCL GENERATION (FULL FUNCTION) ---  
COMMAND ===>  
  
DATE ===> 94/01/10  
TIME ===> 16:33  
  
DBD:          --  
  
PSB NAME =====> _____  
              -  
PCB NUMBER =====> --  
PSB FROM =====> --  
ACBLIB =====> _____  
PSBLIB =====> _____  
RESLIB =====> _____  
  
ENTER PSB NAME, PCB NUMBER, AND INDICATE IF THE ACB OR  
THE PSB LIBRARY IS TO BE USED TO OBTAIN THE PSB FOR THE DATA  
EXTRACTION RUN.  
ENTER AND/OR CORRECT THE DATA SET NAMES AS NEEDED.  
  
PRESS THE 'ENTER' KEY TO CONTINUE THE JCL GENERATION FOR THIS DATA BASE.  
PRESS THE 'END' PFKEY TO TERMINATE THE JCL GENERATION FOR THIS DATA BASE.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is used to specify the information to generate the data sample extraction JCL when the extraction is to be accomplished using DLI in the BATCH environment.

Description of Fields

DBD—DBD name and title of the data base being processed.

PSB NAME—The name of the PSB to be used during the data sample extraction.

PCB NUMBER—The number of the data PCB to use when extracting the data sample. Be sure to allow for the I/O PCB if there is one.

PSB FROM—Allows user to indicated whether the PSB is to be retrieved from the PSBLIB or the ACBLIB at execution time.

ACBLIB—Data set name of the IMS ACBLIB to use in the generated JCL.

PSBLIB—Data set name of the IMS PSBLIB to use in the generated JCL.

RESLIB—Data set name of the IMS RESLIB to use in the generated JCL.

IMS Data Extraction JCL Generation (BMP) Panel - SFEP1JDA

```

SAMS:Compress for IMS -- IMS DATA EXTRACTION JCL GENERATION (BMP) -----
COMMAND ==>

                                                    DATE ==> 94/01/10
                                                    TIME ==> 16:33

DBD:          --
VERIFY AND CORRECT DATA SET NAMES AS NEEDED.

IMS IMSID PARM ==> _____
-
IMS AGN PARM =====> _____
PSB NAME =====> _____
PCB NUMBER =====> ___
IMS RESLIB =====> _____

CORRECT IMS CONTROL REGION ID IF NECESSARY.
ENTER PSB NAME AND PCB NUMBER.
CORRECT THE DATA SET NAME IF NECESSARY.

PRESS THE 'ENTER' KEY TO CONTINUE THE JCL GENERATION FOR THIS DATA BASE.
PRESS THE 'END' PFKEY TO TERMINATE THE JCL GENERATION FOR THIS DATA BASE.
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

```

This panel allows the user to specify the information to generate the data sample extraction JCL when the extraction is to be accomplished using the IMS BMP environment.

Description of Fields

DBD—DBD name and title of the data base being processed.

IMS IMSID PARM—The value to be used for the IMSID execution parameter.

IMS AGN PARM—The value to be used for the AGN execution parameter.

PSB NAME—The name of the PSB to be used for the data extraction.

PCB NUMBER—The number of the PCB to be used for the data extraction. Be sure to include the I/O PCB in the count if there is one.

IMS RESLIB—The data set name of the IMS RESLIB that is to be used for the data extraction JOB.

IMS Data Extraction JCL Generation Panel, Part 2 - SFEP1JEF

```
SAMS:Compress for IMS -- IMS DATA EXTRACTION JCL GENERATION, PART 2 -----
COMMAND ==>

                                         DATE ==> 94/01/10
                                         TIME ==> 16:33

DBD:           --

ENTER EXTRACT FILE ALLOCATION DATA FOR SORTED EXTRACT FILE:
EXT FILE DSN ==> _____

EXT FILE MEDIA ==> _           T = TAPE, D = DASD
DEVICE TYPE ==> _____

DASD ALLOCATION PARAMETERS FOR UNSORTED AND SORTED EXTRACT FILES:
ALLOC UNIT ==> _____      CYL OR TRK
ALLOCATION:  PRIMARY ==> _____ SECONDARY ==> _____

SORT WORK DATA SET ALLOCATION (ALL ALLOCATIONS ARE IN CYLINDRS):
NUMBER OF SORT WORKS ==> 04
ALLOCATION:  PRIMARY ==> 010     SECONDARY ==> 010

PRESS THE 'ENTER' KEY TO CONTINUE JCL GENERATION FOR THIS DATA BASE.
PRESS THE 'END' PFKEY TO TERMINATE JCL GENERATION FOR THIS DATA BASE.
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed for all data extraction jobs. It enables the user to define the data set that will contain the sorted data sample. Also, it allows the user to specify how much sort work space is to be provided.

Description of Fields

DBD—DBD name and title of data base being processed.

EXT FILE DSN—Data set name of the data sample extract file to be created.

EXT FILE MEDIA—The media upon which the data set will be created.

DEVICE TYPE—The logical unit parameter for the DD statement which will define the extracted data sample file.

ALLOC UNIT—DASD unit to use to allocate the temporary and final extracted data sample files.

ALLOCATION—The following two items are under this heading.

PRIMARY—Primary allocation value for extracted data sample file.

SECONDARY—Secondary allocation value for extracted data sample file.

NUMBER OF SORT WORKS—Number of sort work data sets to allocate.

ALLOCATION—The following two items are under this heading.

PRIMARY—Primary allocation value for sort work data sets.

SECONDARY—Secondary allocation value for sort work data sets.

IMS Data Base Implementation Menu- SFEP1IMN

```
SAMS:Compress for IMS -- IMS DATA BASE IMPLEMENTATION MENU -----  
COMMAND ===>  
  
DATE ===> 94/01/10  
TIME ===> 16:33  
  
DBD NAME ===> _____  
-  
  
OPTION ===> _  
1 -- SELECT SEGMENTS FOR COMPRESSION.  
2 -- GENERATE MODIFIED DBD SOURCE MACROS.  
X -- RETURN TO THE IMS SERVICES MENU.  
  
ENTER THE NAME OF THE DBD TO BE PROCESSED. ENTER THE CODE FOR  
THE FUNCTION TO BE PERFORMED. THEN PRESS THE 'ENTER' KEY.  
  
FDTS WILL BE PLACED IN  
SOURCE MACROS WILL BE PLACED IN  
  
GENERATE ALL OF THE FDTS FOR A DATA BASE BEFORE GENERATING THE SOURCE MACROS.  
  
PRESSING THE 'END' PFKEY IS THE SAME AS SELECTING THE 'X' OPTION.  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed when the user elects to implement a data base under the control of SAMS:Compress. The user must identify the data base by its DBD name. Users should first perform function '1', which allows them to select the segments to be compressed. Next they must select option '2' to have the modified DBDs generated.

Description of Fields

DBD NAME—DBD name of the data base to be processed.

OPTION—Type of processing to perform.

FDTS WILL BE PLACED IN—Data set name of the FDT load library.

SOURCE MACROS WILL BE PLACED IN—Data set name of the card image output library.

IMS Segment Selection Panel - SFEP1FDT

```

SAMS:Compress for IMS -- IMS SEGMENT SELECTION -----
COMMAND ==>

DBD NAME:                                     DATE ==> 94/01/10
DBD TITLE:                                    TIME ==> 16:34

SEGMENT SELECTION OPTIONS:

F - CUSTOM HUFFMAN COMPRESSION
E - EXPRESS COMPRESSION
C - CANCEL COMPRESSION

SEL  SEG NAME  FDT SHR  COMMENTS
---  -
-
-

```

This panel is displayed so the user can select the segments to be compressed and to specify the type of compression to be perform for each segment.

Description of Fields

DBD NAME—DBD name of the data base being processed.

DBD TITLE—Title of the data base being processed.

SEL—Segment compression indicator field.

SEG NAME—Segment name.

FDT SHR—FDT share option.

COMMENTS—Segment/FDT status comments.

IMS DBD Generation Panel - SFEP1DBD

```
SAMS:Compress for IMS -- IMS DBD GENERATION -----  
COMMAND ==>  
  
DATE ==> 94/01/10  
TIME ==> 16:34  
  
SOURCE IMSLIB =====> _____  
-  
SOURCE DBD NAME ==> _____  
  
SAMS:Compress DBD NAME ==> _____  
REORG DBD NAME =====> _____  
  
SPECIFY THE CORRECT LIBRARY NAME FOR THE DBD SOURCE AND THE MEMBER  
NAME FOR THE INPUT DBD SOURCE STATEMENTS.  
  
SPECIFY THE MEMBER NAME TO BE USED TO STORE THE NEW SOURCE DBD FOR  
USE BY THE SAMS:Compress ROUTINES.  
  
IF A SEPARATE DBD IS DESIRED FOR REORGANIZATION OF THE COMPRESSED  
DATA, ENTER THE MEMBER NAME TO BE USED TO STORE THIS DBD SOURCE.  
  
PRESSING THE 'END' PFKEY WILL RETURN YOU TO THE IMS SUPPORT SERVICES MENU.  
  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed when it is time to generate the DBDs that allow SAMS:Compress to function with the data base. Through this panel the user informs the Dialog where the current DBD source macros can be located and which member names are to be used by the Dialog as it creates the new DBD macro sets.

Description of Fields

SOURCE IMSLIB—Data set name of the library where the IMS DBD macros will be found.

SOURCE DBD NAME—Member name for this data base's DBD macros.

SAMS:Compress DBD NAME—Member name that is to be used to store the modified DBD macros that will support SAMS:Compress processing.

REORG DBD NAME—Member name that is to be used to store the modified DBD macros that will permit the reorganization of the data base without requiring the expansion and compression of every segment in the data base.

IMS Delete Data Base From the Control File Panel - SFEP1DEL

```
SAMS:Compress for IMS -- DELETE DATA BASE FROM THE DIALOG CONTROL FILE -----  
COMMAND ==>>  
  
DATE ==>> 94/01/10  
TIME ==>> 16:34  
  
DBD NAME ==>> _____  
-
```

ENTER THE DBD NAME OF THE IMS DATA BASE WHICH IS TO BE DELETED.
PRESS THE 'END' PFKEY TO RETURN TO THE IMS SUPPORT SERVICES MENU.

----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----

This panel is displayed when the user elects to delete obsolete data base information from the Interactive Dialog Control File. The panel allows the user to identify the DBD to be deleted.

Description of Field

DBD NAME—DBD name of the data base whose information is to be deleted from the Interactive Dialog Control File.

IMS DBD Data Deletion Confirmation Panel - SFEP1DDC

```
SAMS:Compress for IMS -- IMS DBD DATA DELETION CONFIRMATION -----  
COMMAND ==>>  
  
DATE ==>> 94/01/10  
TIME ==>> 16:35  
  
DBD NAME:  
DATA BASE TITLE:  
IMS GEN DATE:          TIME:          RELEASE:  
  
CREATED BY:          DATE:          TIME:  
  
CONFIRM DELETE ==>> _      'N' -- DO NOT DELETE DATA BASE DATA  
                   -      'Y' -- DELETE DATA BASE DATA  
  
IF THIS IS NOT THE DBD TO BE DELETED, PRESS THE ENTER KEY.  
  
IF THIS IS THE DBD TO BE DELETED, ENTER A 'Y' IN THE CONFIRM DELETE  
FIELD AND PRESS THE ENTER KEY.  
  
PRESSING THE 'END' PFKEY IS THE SAME AS PRESSING THE 'ENTER' KEY  
WITHOUT CHANGING THE CONFIRM DELETE FIELD.  
  
----- PRESS THE 'HELP' PFKEY FOR INSTRUCTIONS ABOUT THIS PANEL -----
```

This panel is displayed so the user can verify the data base that is about to be deleted and confirm whether it is the correct data base to delete.

Description of Fields

DBD NAME—DBD name of data base to be deleted.

DATA BASE TITLE—Title of data base to be deleted.

IMS GEN DATE—DBD generation date.

TIME—DBD generation time.

RELEASE—Release of IMS used to generate the DBD.

CREATED BY—SAMS:Compress analysis of data base perform by.

DATE—Date analysis was performed.

TIME—Time analysis was performed.

CONFIRM DELETE—Data base deletion confirmation field.

Chapter 6. SAMS:Compress/2

Overview

The SAMS:Compress/2 Data Compression System provides the means by which your applications programs can invoke subroutines to compress and expand data records used by the applications programs, reducing the cost of storing data on disk or tape. Utilities are provided that will compress or expand the following:

- Entire key-sequenced or entry-sequenced VSAM files,
- Entire sequential or indexed-sequential files,
- Entire direct access files and members of partitioned data sets when being processed sequentially.

SAMS:Compress/2 is provided to all SAMS:Compress users. It is intended to be used without SAMS:Compress for IMS and SAMS:Compress for IMS Express, and without the facilities of the Interactive Dialog.

Features

Executing under OS or OS/VS, SAMS:Compress/2 provides a unique combination of capabilities:

High Degree of Data Compression

Data sets compressed by SAMS:Compress/2 require as little as 10 percent, and commonly 35 percent, of their original space allocation.

Efficient Implementation

Applications using the SAMS:Compress/2 system to process compressed data sets can complete in less time than was once required to process non-compressed data sets.

Demonstrated Reliability

SAMS:Compress/2 uses a redundancy check-byte technique to ensure that no data is altered during compression and expansion.

Ease of Use

SAMS:Compress/2 provides excellent compression without requiring the user to provide information about the structure and content of records to be compressed. For users desiring to achieve compression ratios approaching the theoretical maximum, SAMS:Compress/2's Record Definition Language (RDL) provides the opportunity to describe the record structure and content of the data set to be compressed. In the absence of user-coded RDL specifications, SAMS:Compress/2 assumes standard defaults based upon attributes of the input data set to achieve excellent compression.

Flexibility

The Record Definition Language provides a powerful tool for constructing a compressed data set tailored to the user's specific requirements. User-specified fields may be exempted from compression, allowing sorting of compressed data sets without requiring prior re-expansion. Another advantage is that application programs can expand records selectively (i.e., only those which require further processing) based on an examination of a field or fields exempted from compression. In this manner, unnecessary expansion overhead is avoided.

High Level of Security

SAMS:Compress/2 compresses data by altering the coding structure used to represent the data in a data set. The information concerning the new coding structure is stored in a separate File Descriptor Table (FDT) for each data set compressed. The compressed data set is encrypted according to the information in the FDT, and is unintelligible to unauthorized persons.

SAMS:Compress/2 processes single or multiple compressed data sets in a batch, multitasking or teleprocessing environment.

After compression, data sets almost always contain variable-length records. Since COBOL does not support variable-length ISAM records, SAMS:Compress/2 provides subroutines that enable users to process compressed ISAM data sets via calls from COBOL application programs. These subroutines are invoked using standard CALL conventions.

Modes of Use

There are two ways of using SAMS:Compress/2. Each of these modes is described in more detail in the next two chapters.

Utility Programs

As a group of data set utility programs, SAMS:Compress/2 can be used to compress and expand an entire data set, or a subset of a data set can be compressed to obtain compression statistics. Figure 6-1 shows an overview of the SAMS:Compress/2 utilities.

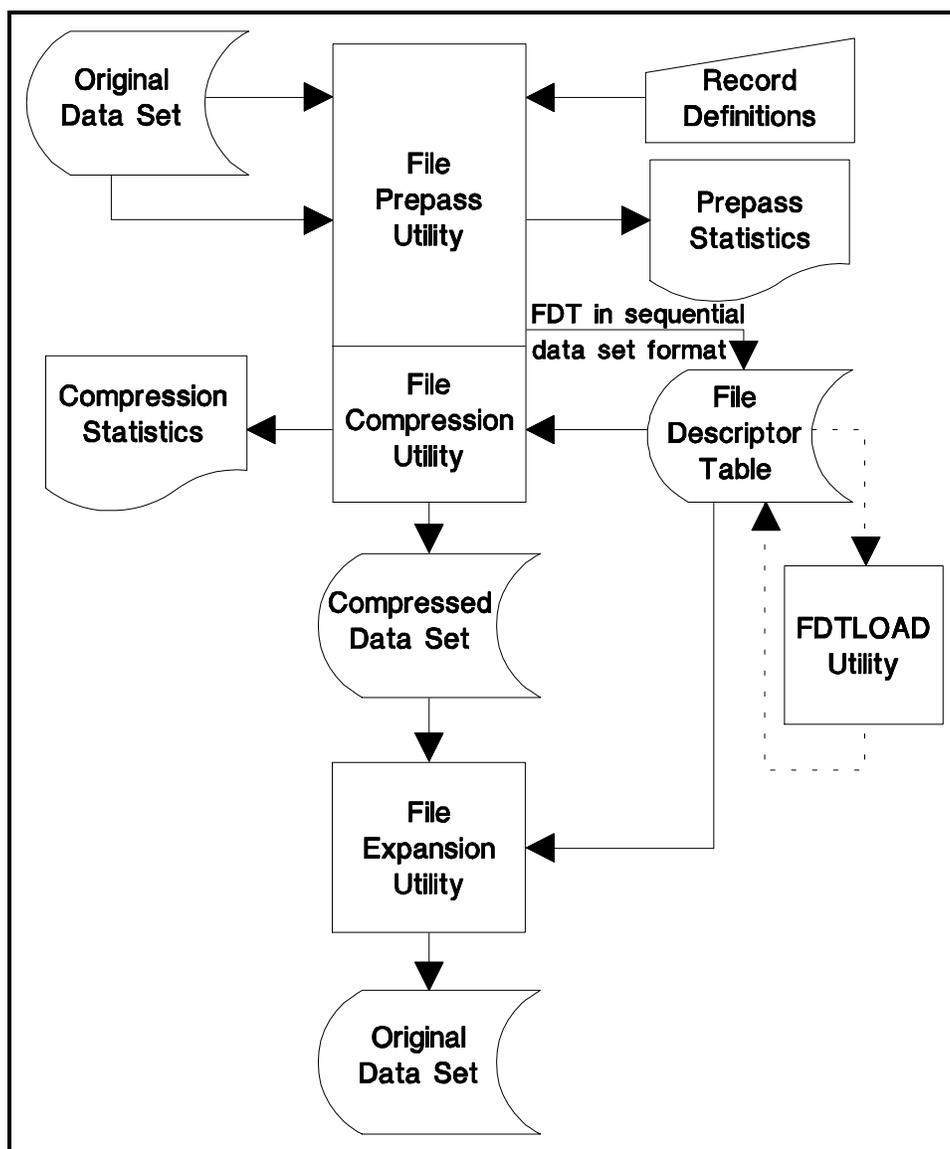


Figure 6-1. SAMS:Compress/2 Operation

Subroutines

As subroutines, SAMS:Compress/2 modules can be called from application programs written in Assembler Language or high-level languages. These modules are invoked to compress and expand records already in storage. The application program performs the I/O. Records in storage can be expanded, updated and compressed.

A typical procedure is to compress a data set once, using the File Compression Utility, then access the compressed records from an application program, using SAMS:Compress/2 system subroutines.

With SAMS:Compress/2, the user has the option of expanding/recompressing each record read, or performing selective expansion/recompression, based on user-specified uncompressed fields within each compressed record. The uncompressed fields may be ISAM keys, sort keys, or any fields which the user chooses to exempt from compression.

Compressed records containing sort key fields which are exempted from compression can be sorted by sort utility programs without requiring prior re-expansion. Similarly, if match key fields are exempted from compression, application programs containing logic to match records based on match key fields can perform the matching logic on the compressed record(s) and limit record expansion to only those cases which require further processing.

The Record Definition Language allows the user to exempt fields from compression as well as to optimize compression. Fields in the record are defined to SAMS:Compress/2 by field type code specifications, which indicate the method of compression for the defined field. The set of type codes lets the user define virtually any combination of field formats.

Use of the RDL is optional. If RDL specifications are omitted by the user, SAMS:Compress/2 assumes default RDL specifications based upon data set attributes provided by JCL or by the data set label.

Using the File Utility Programs

Four stand-alone utilities are supplied with SAMS:Compress/2:

- File Prepass Utility
- FDTLOADR Utility
- File Compression Utility
- File Expansion Utility

JCL to execute these utilities is illustrated in this chapter, and is also provided in the SHRINK JCL file. See the Installation chapter for more information on the JCL file.

Before a file can be processed, all or a portion of the file must be “prepassed” using the File Prepass Utility. The File Prepass Utility collects statistical information about compression of the prepassed file and builds a short, 1 to 8Kb, sequential data

set (called the File Descriptor Table, or FDT) which holds this information. The FDT must be present (through JCL specification) in every job step which uses SAMS:Compress/2 functions to operate upon the file.

The FDT can optionally be converted from a sequential data set to a load module by using the FDTLOADR Utility.

After a file is prepassed, all or part of the file is compressed using the File Compression Utility, which also prints compression statistics. Sequential, ISAM, and key-sequenced VSAM files are eligible, as are PDS (partitioned data set) members and direct files, if they are processed sequentially.

A compressed file can be returned to its original uncompressed form using the File Expansion Utility.

These utilities are not re-entrant and may not reside in the Link Pack area. They must be defined through STEPLIB or JOBLIB in the JCL.

File Prepass Utility

Files processed by SAMS:Compress/2 must be initially defined by executing the SAMS:Compress program with the prepass option, which performs the following:

- Reads and edits any record definitions supplied by the user.
- Creates default record definitions if the user did not specify them.
- Reads all or part of the file to collect statistical information relating to compression.
- Constructs and writes the File Description Table as a sequential data set.
- Prints prepass statistics.

File Descriptor Table

The File Description Table, written as a result of executing the File Prepass Utility, is a short, 1 to 8Kb, sequential data set containing all of the information (code tables, edited record definitions, file characteristics, etc.) needed to change the form of the prepassed file from uncompressed to compressed and from compressed to uncompressed. Each file processed must have an FDT. The FDT associated with a particular file must be present, through JCL specification, in every job step in which records from that file are processed by SAMS:Compress/2.

Once a file is compressed, the data contained in the file appears unintelligible since its coding structure is modified by SAMS:Compress/2 during file compression. The FDT contains the information which SAMS:Compress/2 requires to return the file (or individual records from the file) to original uncompressed form. Control of the FDT is equivalent to control of the file. The user can take advantage of this fact as part of the security system to limit access to compressed (and thus, encrypted) files.

The FDT can be converted, at the user's option, to a load module and stored as a member of a load library PDS. Converting FDTs to load modules can reduce JCL coding in application programs using SAMS:Compress/2 subroutines to expand and compress re-

cords. This process is described in the subsection “FDTLOADR Utility” on page 117.

Execution Example—File Prepass Utility

The File Prepass Utility is invoked by executing the program named SHRINK and supplying a PARM value in the form “P=xxx”, where xxx is either the word ALL or a three-digit number. If ALL is coded, the entire data set defined by the INFILE DD statement is prepassed (read). Coding a three-digit number indicates, in thousands, the number of logical records to prepass. The first xxx thousand records in the data set are read, the FDT is constructed and written, and the utility goes immediately to end-of-job without reading the remaining records in the data set.

In the example, the PARM specification “P=015” indicates that the first 15,000 records of the INFILE data set are prepassed (or the entire data set if it contains fewer than 15,000 records). In most cases, it is unnecessary to prepass more than five to ten percent of a data set in order to collect accurate compression statistics for the FDT. If no type C fields are specified in the RDL (or assumed by default), prepassing 1,000 records (“P=001”) is sufficient.

The RECDEF DD statement defines the data set which contains user-specified record definitions describing the file before it is prepassed. As shown, user-specified record definitions are optional. If omitted, the File Prepass Utility assumes default record definitions based upon DCB attributes of the file that is prepassed, as determined from its file label and parameters coded on the INFILE DD statement. Normally, and as shown in the example, record definitions are submitted in-stream, though they may be contained in a PDS member or a separate sequential data set at the user’s option. The default definitions assumed by the File Prepass Utility produce satisfactory compression and often execute faster than more precise definitions specified by the user.

The Record Definition Language chapter contains a complete description of field types, user-specified record definitions and default record definitions.

COBOL users should specify record definitions in all cases. To direct the File Prepass Utility to supply default definitions for COBOL users, the following minimum record definition should be supplied:

```
//RECDEF DD *  
L.  
/*
```

Since compressed records are almost always variable-length records, users who follow this suggestion find it easier to process variable-length compressed records in COBOL application programs. Users may optionally specify more detailed definitions in lieu of the default definitions, which the File Prepass Utility supplies in response to the specification suggested here. Special considerations for defining compressed records in COBOL application programs are discussed later in this chapter.

Figure 6-2 defines the cataloged data set, “user.input.file”, to the SAMS:Compress/2 system and creates its associated File Descriptor Table, “user.fdt.file”.

```

/*SHR2PASS JOB
/*
/*
/******
/*
/* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 USERS TO *
/* BUILD THEFILE DESCRIPTOR TABLE (FDT) *
/* *
/******
//PREPASS EXEC PGM=SHRINK,PARM='P=015'
//STEPLIB DD DISP=SHR, DSN=user.shrink.load
//PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE DD DISP=SHR,DSN=user.input.file
//TABL00 DD DISP=(,CATLG),DSN=user.fdt.file,
UNIT=unit,VOL=SER=volser,SPACE=(2048,(3,1))
//RECDEF DD *
record definitions (optional)
/*
//

```

Figure 6-2. SHRINK File Prepass Utility Job Step

Prepass Statistics

The printed output produced by the File Prepass Utility comprises the following:

- The input file’s DCB information (RECFM, LRECL, BLKSIZE, and, if applicable, RKP and KEYLEN).
- Record definitions (RDL statements) either specified by the user or assumed by default.
- Error messages indicating incorrect usage of the Record Definition Language. Each message is prefixed by an alphanumeric character, which also underscores the error in the RDL statement to which the message applies. See the Messages and Abend Codes chapter for details.
- The number of records prepassed.
- The shortest, longest, and average lengths of records contained on the prepassed file. For fixed-length records, all three numbers are identical.

Furthermore, if field types C1, C2 and/or C3 are defined in the RDL for the file that is prepassed (either by user specification or by default), a table is printed for each such field type defined. This table shows the post-compression bit code for each of the 256 possible values that can be contained in one byte, plus the bit code used to represent the repetition indicator. The post-compression bit codes are assigned by the File Prepass Utility based on the relative frequency of occurrence of each byte value encountered during prepass; most frequently encountered values are assigned the shortest post-compression bit codes, and least frequently encountered values are assigned the longest post-compression bit codes.

Figure 6-3 below shows a sample printout of a field type C table.

```

SAMS:Compress -- Sterling Software, Inc.
INFILE DCB: RECFM=FB  BLKSIZE=18960  LRECL=00120
INFILE DSNNAME=SBMHM.SHR.TSTFLE05
CIVER.
000001217 RECORDS PREPASSED.  SHORTEST=00120  LONGEST=00120  AVERAGE=00120

TYPE-C1
V FREQUENCY LN CODE  V FREQUENCY LN CODE  V FREQUENCY LN CODE  V FREQUENCY LN CODE  V FREQUENCY LN CODE
00      1 16 E0C0  34      1 16 E0C6  68      1 15 4564  9B      1 16 E0BF  CE      1 16 E0C9
01      1 15 4580  35      1 15 458C  69      1 15 45C4  9C      1 15 45EE  CF      1 15 4592
02      1 16 E0E0  36      1 16 E0E6  6A      1 15 4544  9D      1 16 E0B7  D0      1 16 E0E9
03      1 15 4540  37      1 15 454C  6B      488 06 38  9E      1 15 452E  D1      182 08 96
04      1 15 45C0  38      1 15 45CC  6C      1 16 E0E2  9F      1 16 E0D7  D2      459 06 08
05      1 15 4560  39      1 15 456C  6D      1 15 4584  A0      1 15 45AE  D3      871 06 E4
06      1 16 E0F0  3A      1 16 E0F6  6E      1 16 E0C2  A1      1 16 E0F7  D4      829 06 BC
07      1 15 45A0  3B      1 15 45AC  6F      1 15 4504  A2      1 15 456E  D5      1562 05 B0
08      1 16 E0D0  3C      1 16 E0D6  70      1 16 45FE  A3      1 15 45CE  D6      1361 05 80
09      1 15 4520  3D      1 15 452C  71      1 15 45FC  A4      1 15 454E  D7      716 06 90
0A      1 16 E0B0  3E      1 16 E0B6  72      1 16 45FF  A5      1 16 E0E7  D8      5 13 E0A0
0B      1 15 45E0  3F      1 15 45EC  73      1 15 4506  A6      1 15 458E  D9      1221 05 50
0C      1 16 E0B8  40      6329 03 C  74      1 16 E0C3  A7      1 16 E0C7  DA      1 15 4552
0D      1 15 4530  41      1 16 E0BE  75      1 15 4586  A8      1 15 450E  DB      1 15 45D2
0E      1 16 E0D8  42      1 15 453C  76      1 16 E0E3  A9      1 15 45FA  DC      1 15 4572
0F      1 15 45B0  43      1 16 E0DE  77      1 15 4546  AA      1 15 450A  DD      1 16 E0F9
10      1 16 E0F8  44      1 15 45BC  78      1 15 45C6  AB      1 16 E0C5  DE      1 15 45B2
11      1 15 4570  45      1 16 E0FE  79      1 15 4566  AC      1 15 458A  DF      1 16 E0D9
12      1 15 45D0  46      1 15 457C  7A      1 16 E0F3  AD      1 16 E0E5  E0      1 15 4532
13      1 15 4550  47      1 15 45DC  7B      210 08 E1  AE      1 15 454A  E1      1 16 E0B9
14      1 16 E0E8  48      1 15 455C  7C      1 15 45A6  AF      1 15 45CA  E2      2522 04 6
15      1 15 4590  49      1 16 E0EE  7D      44 10 E04  B0      1 15 456A  E3      1386 05 88
16      1 16 E0C8  4A      1 15 459C  7E      517 06 3C  B1      1 16 E0F5  E4      433 06 00
17      1 15 4510  4B      601 06 4C  7F      1 16 E0D3  B2      1 15 45AA  E5      211 07 04
18      1 15 45F0  4C      1 16 E0CE  80      1 15 4526  B3      1 16 E0D5  E6      244 07 06
19      1 15 4518  4D      191 08 B8  81      1 16 E0B3  B4      1 15 452A  E7      309 07 46
1A      1 16 E0CC  4E      1 15 451C  82      1 15 45E6  B5      1 16 E0B5  E8      394 07 BA
1B      1 15 4598  4F      1 15 45F4  83      1 16 E0BB  B6      1 15 45EA  E9      27 11 E08
1C      1 16 E0EC  50      43 10 E00  84      1 15 4536  B7      1 16 E0BD  EA      1 15 45E2
1D      1 15 4558  51      1 15 4514  85      1 16 E0DB  B8      1 15 453A  EB      1 16 E0B1
1E      1 15 45D8  52      1 16 E0CA  86      1 15 45B6  B9      1 16 E0DD  EC      1 15 4522
1F      1 15 4578  53      1 15 4594  87      1 16 E0FB  BA      1 15 45BA  ED      1 16 E0D1
20      1 16 E0FC  54      1 16 E0EA  88      1 15 4576  BB      1 16 E0FD  EE      1 15 45A2
21      1 15 45B8  55      1 15 4554  89      1 15 45D6  BC      1 15 457A  EF      1 16 E0F1
22      1 16 E0DC  56      1 15 45D4  8A      1 15 4556  BD      1 15 45DA  F0      1854 04 1
23      1 15 4538  57      1 15 4574  8B      1 16 E0EB  BE      1 15 455A  F1      1269 05 58
24      1 16 E0BC  58      1 16 E0FA  8C      1 15 4596  BF      1 16 E0ED  F2      647 06 78
25      1 15 45E8  59      1 15 45B4  8D      1 16 E0CB  C0      1 15 459A  F3      326 07 76
26      1 16 E0B4  5A      1 16 E0DA  8E      1 15 4516  C1      1738 05 E8  F4      624 06 70
27      1 15 4528  5B      2 14 4500  8F      1 15 45F6  C2      463 06 0C  F5      652 06 7C
28      1 16 E0D4  5C      565 06 40  90      1 15 451E  C3      993 05 30  F6      182 08 97
29      1 15 45A8  5D      191 08 B9  91      1 16 E0CF  C4      1518 05 A8  F7      179 08 95
2A      1 16 E0F4  5E      1 15 4534  92      1 15 459E  C5      1889 04 2  F8      176 08 94
2B      1 15 4568  5F      1 16 E0BA  93      1 16 E0EF  C6      424 07 E2  F9      321 07 74
2C      1 15 45C8  60      7 13 E0A8  94      1 15 455E  C7      129 08 44  FA      1 15 4562
2D      1 15 4548  61      1451 05 98  95      1 15 45DE  C8      568 06 48  FB      1 15 45C2
2E      1 16 E0E4  62      1 15 45E4  96      1 15 457E  C9      1462 05 A0  FC      1 15 4542
2F      1 15 4588  63      1 16 E0B2  97      1 16 E0FF  CA      1 16 E0CD  FD      1 16 E0E1
30      1 16 E0C4  64      1 15 4524  98      1 15 45BE  CB      1 15 451A  FE      1 15 4582
31      1 15 4508  65      1 16 E0D2  99      1 16 E0DF  CC      1 15 45F2  FF      1 16 E0C1
32      1 15 45F8  66      1 15 45A4  9A      1 15 453E  CD      1 15 4512  RP      3463 04 F
33      1 15 450C  67      1 16 E0F2

OUTFILE DCB: RECFM=VB  BLKSIZE=18960  LRECL=00128

74 PER CENT AVERAGE REDUCTION

AFTER COMPRESSION: 0000001217 RECORDS ,00000000037096 BYTES.  SHORTEST=00010  LONGEST=00056  AVERAGE=00030
BEFORE COMPRESSION: 0000001217 RECORDS ,00000000146040 BYTES.  SHORTEST=00120  LONGEST=00120  AVERAGE=00120
    
```

Figure 6-3. Field Type C Table

The values in this table are:

V	Uncompressed byte value in hexadecimal.
FREQUENCY	The number of times the byte value occurred outside of a repetition string.
LN	The length, in bits, of the post-compression bit code.
CODE	The post-compression bit code in hexadecimal, padded on the right with zeros.

RP is the repetition indicator, and it appears as the last entry in the printed table. The frequency shown for the repetition indicator denotes the number of strings of three or more consecutive identical bytes detected while compressing all fields included in this frequency table (e.g., all C1 fields).

The printed output of the File Prepass Utility is a valuable record of the conversion of an uncompressed file to compressed format, and it should be retained for future reference.

FDTLOADR Utility

The user may wish to convert the FDT(s) to load module format. This conversion allows the user to group FDTs within a partitioned data set (PDS); each FDT is a separately named PDS member.

Advantages of Use of FDTs in Load Module Format

Conversion of FDTs to load module format offers the following advantages:

- The need for a separate TABLxx DD statement for each FDT used in an application program is eliminated.
- Only a single STEPLIB DD statement identifying the PDS containing all of the FDTs used in a job step is required.

Thus, JCL coding is reduced in application programs which process more than one compressed data set, and the storage of FDTs is centralized for ease of control and maintenance.

FDTs in load module format are accessed from application programs in a slightly different manner than FDTs in sequential data set format. Thus, converting existing FDTs to load module format requires minor modifications to application programs which are already coded to process FDTs in sequential data set format. (See “Accessing the FDT” later in this chapter.)

The SYSLMOD DD statement defines the PDS library and the member name in which the FDT is stored in load module format. The example presumes that the PDS already exists and is cataloged. Substitute the name of the PDS for

“user.fdt.load”); substitute the desired member name of the FDT in load module format for “fdtname”.

FDT member names must be unique within a PDS library containing FDTs. It is strongly recommended that FDT member names be unique throughout all PDS libraries containing FDTs.

The TABL00 DD statement defines the FDT in sequential data set format for conversion to load module format. Substitute the correct data set name for “user.fdt.file” as it was coded on the TABL00 DD statement used in the execution of the File Prepass Utility which created the FDT. The example presumes that the FDT in sequential data set format was cataloged when created and should be deleted upon conversion to load module format.

Execution Example—FDTLOADR Utility

Figure 6-4 illustrates the JCL necessary to invoke the FDTLOADR Utility to convert one FDT from sequential data set format to load module format.

```

//*SHR2FLDR JOB
//*
//*
//*****
//*
//* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 AND *
//* SAMS:COMPRESS FOR IDMS USERS TO CONVERT A SEQUENTIAL FDT TO *
//* LOAD MODULE FORMAT *
//*
//*****
//CONVFDT EXEC PGM=FDTLOADR
//STEPLIB DD DISP=SHR,DSN=user.shrink.load
//SYSLIN DD UNIT=SYSDA,SPACE=(TRK,(2,1))
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=user.fdt.load(fdtname)

```

Figure 6-4. FDTLOADR Utility Job Step

File Compression Utility

Once a file is prepassed and the FDT created, the file may be compressed using either the File Compression Utility or a user program that calls the SHRINK subroutine. Common practice is to use the File Compression Utility for two reasons: No user coding is required (beyond JCL), and the File Compression Utility automatically produces (and prints) compression statistics. The SAMS:Compress/2 subroutine does not produce compression statistics.

Considerations Before Compressing

Coding the DSORG parameter on the OUTFILE DD statement is required for ISAM files with DISP=(NEW,...). Coding the primary and secondary space allocations, “(p,s)”, depends on several factors:

1. The total number of blocks required by the compressed data set can be approximated using the following formula:

$$p = \text{Average } L/b * NR$$

where:

p = The number of blocks required of BLKSIZE b.

b = BLKSIZE of OUTFILE data set.

Average L = The average compressed record length from compression statistics. (See “Compression Statistics” on page 121.)

NR = The number of logical records contained in OUTFILE data set.

2. The values chosen for primary and secondary allocations depend upon the availability of space at the user’s installation, the degree of fragmentation which the user finds acceptable, and the expected growth (or shrinkage) of the data set over time. Ideally, the user is able to allocate the expected number of blocks as calculated above as the primary allocation. The secondary allocation depends on the expected change in size of the data set over time.

The File Compression Utility assumes default values for certain OUTFILE DCB parameters if they are not specified by the user. The DSORG parameter must be specified only for ISAM files with DISP=(NEW,...). The RECFM, LRECL, BLKSIZE, and, if applicable, RKP and KEYLEN parameters can be omitted in most cases. The table shown on the next page is provided to assist you in determining the LRECL for the compressed data set.

The LRECL value should be at least eight bytes greater than the INFILE LRECL. To the LRECL of the uncompressed data set, add (or subtract, in the case of type GA) the following values:

Table 6-1. Calculation of LRECL for Compressed Data Sets

When	How Much
Always	+2
Type L defined	+2
“VER” length descriptor defined	+2
RDW present; i.e., not RECFM=U, F, or VSAM	+4
Each type GA definition	-field length
Each non-GA field redefined via the Position function	+field length

Note: For each condition group series defined, use the length of the condition group which generates the largest LRECL by using the above table.

To ensure data integrity, a redundancy check byte is calculated and appended to each compressed record, no matter which compression method is used.

The File Compression Utility is invoked by executing the program named SHRINK and by supplying a PARM value in the form C=xxx, where xxx is either the word ALL or a three-digit number. If ALL is coded, the entire data set defined by the INFILE DD statement is compressed. Coding a three-digit number indicates, in thousands, the number of logical records to compress from the data set defined by the INFILE DD statement. The first xxx thousand records in the data set are compressed, and then the File Compression Utility goes to end-of-job.

For the first attempt to compress a very large data set, it is better to code a three-digit number to compress a subset of the data set as a test run, rather than to attempt to compress the entire data set. Thus, the user is given an opportunity to evaluate any warning conditions which may arise and to take corrective action. (See the Messages and Abend Codes chapter.)

Execution Example—File Compression Utility

Figure 6-5, below, illustrates the compression of the entire file, USER.FILE.

```

/*SHR2CMP1 JOB
/*
/*
/******
/*
/* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 USERS TO      *
/* PERFORM COMPRESSION OF A SPECIFIED DATA SET                      *
/******
/*COMPRESS EXEC PGM=SHRINK,PARM='C=ALL'
/*STEPLIB DD DISP=SHR,DSN=user.shrink.load
/*PRINT DD SYSOUT=*
/*SYSUDUMP DD SYSOUT=*
/*INFILE DD DISP=SHR,DSN=user.input.file
/*OUTFILE DD DISP=(,CATLG),DSN=user.comprsd.file,
UNIT=unit,VOL=SER=volser,SPACE=(b,(p,s))
/*TABL00 DD DISP=SHR,DSN=user.fdt.file

```

Figure 6-5. File Compression Utility—FDT in Sequential Data Set Format

File Compression Utility With FDT in Load Module Format

The example shown in Figure 6-5 assumes the use of an FDT in sequential data set format. The user may choose to execute the File Compression Utility using an FDT in load module format. To do so, the library containing FDT in load module format must be that library defined by the STEPLIB DD statement (or concatenated with it), the TABL00 DD statement must be omitted, and the member name identifying the FDT in load module format must be specified in the form FN=fdname as a

PARM parameter on the EXEC statement. Figure 6-6 shows the example presented in Figure 6-5, modified to use an FDT in load module format.

```

/*SHR2CMP2 JOB
/*
/*
/******
/*
/* THE FOLLOWING JCL IS PROVIDED FOR SHRINK/2 USERS TO PERFORM *
/* COMPRESSION OF A SPECIFIED DATA SET *
/*
/******
//COMPRESS EXEC PGM=SHRINK,PARM='C=ALL, FN=fdtname'
//STEPLIB DD DISP=SHR,DSN=user.shrink.load
// DD DISP=SHR,DSN=user.fdt.load
//PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE DD DSP=SHR,DSN=user.input.file
//OUTFILE DD DISP=(,CATLG),DSN=user.comprsd.file,
UNIT=unit,VOL=SER=volser,SPACE=(b,(p,s))

```

Figure 6-6. File Compression Utility—FDT in Load Module Format

Compression Statistics

The File Compression Utility directs a printed report to the data set defined by the PRINT DD statement. This report contains the following information:

The DCB information (RECFM, LRECL, BLKSIZE, and, if applicable, RKP and KEYLEN) of a sequential or ISAM data set, or the VSAM information (LRECL, cluster type, and, if applicable, the RKP and KEYLEN), whether specified by the user or assumed by default, for both the data set defined by the INFILE DD statement and the data set defined by the OUTFILE DD statement.

The data set name for both the INFILE DD statement and the data set defined by the OUTFILE DD statement. This will not be printed if the respective DD statement is a dummy data set.

The number of records compressed and the number of bytes compressed and produced.

Note: The amount of disk or tape storage media required to contain this amount of data also depends on other factors not under the control of SAMS:Compress/2.

The shortest, longest and average record lengths, both before and after compression.

Note: The compressed length includes the four-byte RDW, if present.

The average compression percentage defined as follows, which shows that 0 indicates no compression, and 100 indicates total compression.:

$$100 * \frac{X-Y}{X}$$

where:

X = average record length before compression.

Y = average record length after compression.

The number of type PD, ZL, ZR, S, and X fields which contained information incompatible with the field definition. (See Record Definition Language chapter for more information on these field types.)

Figure 6-7 shows an example of compression statistics produced by the File Compression Utility.

```
INFILE DCB: RECFM=FB  BLKSIZE=18960  LRECL=00120
INFILE  DSNAME=SBMHM.SHR.TSTFLE05
CLVER.
000001217 RECORDS PREPASSED. SHORTEST=00120  LONGEST=00120  AVERAGE=00120
OUTFILE DCB: RECFM=VB  BLKSIZE=18960  LRECL=00128
74 PER CENT AVERAGE REDUCTION
AFTER  COMPRESSION: 000001217 RECORDS ,00000000037096 BYTES.  SHORTEST=00010  LONGEST=00056  AVERAGE=00030
BEFORE COMPRESSION: 000001217 RECORDS ,00000000146040 BYTES.  SHORTEST=00120  LONGEST=00120  AVERAGE=00120
```

Figure 6-7. Compression Statistics Produced by the File Compression Utility

Combining File Prepass and File Compression in a Single Job Step

Both the File Prepass Utility and the File Compression Utility are contained in the program named SHRINK. As described earlier, the utility invoked for a particular execution of the SHRINK program depends on a PARM value coded by the user on the EXEC statement. The user may optionally invoke both utilities in a single job step by specifying “P=xxx,C=xxx” (where xxx is the word ALL or a three-digit number) for the PARM value. The File Prepass Utility executes first, as directed by the P=xxx PARM value. Then the File Compression Utility executes, as directed by the C=xxx PARM value. This option is not available if the INFILE DD statement defines an in-stream data set (INFILE DD *). Figure 6-8 presents an example of this procedure.

```

/*PASSCOMP JOB
/*
/*
/******
/* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 USERS TO *
/* BUILD THEFILE DESCRIPTOR TABLE (FDT) AND PERFORM TRIAL COMPRESS*
/* STATISTICS *
/*
/******
/*PASSCOMP EXEC PGM=SHRINK,PARM='P=010,C=005
/*STEPLIB DD DISP=SHR,DSN=user.shrink.load
/*PRINT DD SYSOUT=*
/*SYSUDUMP DD SYSOUT=*
/*INFILE DD DISP=SHR,DSN=user.input.file
/*OUTFILE DD DUMMY
/*TABL00 DD DISP=(,CATLG),DSN=user.fdt.file,
UNIT=unit,VOL=SER=volser,SPACE=(2048,(3,1))
/*RECDEF DD *
record definitions (optional)
/*

```

Figure 6-8. Combined Execution of Prepass and Compression Utilities

FDTs in load module format cannot be processed in a job step that invokes both the File Prepass Utility and the File Compression Utility. In other words, coding P=xxx and FN=fdname as PARM values in the same job step is not permitted. The FDT in sequential data set format produced from this job step can be run through the FDTLOADR Utility in a separate job step to produce the FDT in load module format. Both prepass and compression statistics are produced.

This example prepasses the first 10,000 records (P=010) from user.input.file, produces prepass statistics, and creates the FDT, user.fdt.file. Then the first 5,000 records (C=005) of user.input.file are compressed, producing compression statistics.

The compressed output is discarded (DUMMY parameter on OUTFILE DD statement). The purpose of executing a job step such as this is to effect a trial compression, to make sure that record definitions are correct and provide adequate compression, to obtain timing estimates (if desired), and to obtain compression statistics from which compressed data set space requirements can be estimated.

File Expansion Utility

The File Expansion Utility expands a data set previously compressed by either the File Compression Utility or via user program calls to subroutine SHRINK. The entire data set is expanded. The expanded data set is identical to the data set as it existed prior to compression. Data integrity is ensured by comparison of the redundancy check byte appended to each compressed record when the data set was compressed, with a redundancy check byte calculated as each record is expanded. The File Expansion Utility prints the INFILE and OUTFILE DCB information as in the File Compression Utility.

Execution Example—File Expansion Utility

The File Expansion Utility is invoked by executing the program named EXPAND. Unlike the File Compression Utility, the File Expansion Utility does not offer the option to process only a portion of the input data set.

The OUTFILE DD statement defines the uncompressed data set written by the File Expansion Utility. The RECFM DCB attribute must be the same as the RECFM DCB attribute of the original uncompressed data set. So must the LRECL DCB attribute be the same; however, the number of records per block may differ. The values coded for primary and secondary space allocations can be those of the original uncompressed file provided that the file has not grown (through normal update processing) since it was compressed. Otherwise, the user must calculate appropriate values, based upon the original file's BLKSIZE, LRECL, number of records in the file being expanded, and the device type to which the expanded data set is written.

The File Expansion Utility assumes default values for certain DCB parameters if they are not specified by the user. The DSORG parameter must be specified only for ISAM files with DISP=(NEW,...). The RECFM, LRECL, BLKSIZE, and, if applicable, RKP and KEYLEN parameters can be omitted in most cases. A full description of default DCB parameters is found following this example. Figure 6-9 illustrates the expansion of the entire file, USER.FILE.

```

//*SHR2EXP1 JOB
//*
//*
//*****
//*
//* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 USERS TO *
//* PERFORM EXPANSION OF A SPECIFIED DATA SET *
//* *
//*****
//EXPAND EXEC PGM=EXPAND
//STEPLIB DD DISP=SHR,DSN=user.shrink.load
//PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE DD DISP=SHR,DSN=user.comprsd.file
//OUTFILE DD DISP=(,CATLG),DSN=user.output.file
           UNIT=unit,VOL=SER=volser,SPACE=(b,(p,s))
//TABL00 DD DISP=SHR,DSN=user.fdt.file

```

Figure 6-9. File Expansion Utility—FDT in Sequential Data Set Format

File Expansion Utility With FDT in Load Module Format

The example shown in Figure 6-9 assumes the use of an FDT in sequential data set format. The user may choose to execute the File Expansion Utility using an FDT in load module format. To do so, the library containing the FDT in load module format must be that library defined by the STEPLIB or JOBLIB DD statement (or concatenated with it), the TABL00 DD statement must be omitted, and the member name identifying the FDT in load module format must be specified in the form FN=fdname as a PARM parameter on the EXEC statement.

Figure 6-10 shows the example in Figure 6-9, modified to use an FDT in load module format.

```

//*SHR2EXP2 JOB
//*
//*
//*****
//*
//* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS/2 USERS TO *
//* PERFORM EXPANSION OF A SPECIFIED DATA SET *
//* *
//*****
//EXPAND EXEC PGM=EXPAND,PARM='FN=fdname'
//STEPLIB DD DISP=SHR,DSN=user.shrink.load
//PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE DD DISP=SHR,DSN=user.comprsd.file
//OUTFILE DD DISP=(,CATLG),DSN=user.output.file,
UNIT=unit,VOL=SER=volser,SPACE=(b,(p,s))

```

Figure 6-10. File Expansion Utility—FDT in Load Module Format Pass the data address

Default JCL for the Compression and Expansion Utilities

Certain DCB parameters for the OUTFILE data set of the Compression and Expansion Utilities need not be specified explicitly by the user via JCL; certain reasonable defaults are assumed. Only sequential, ISAM, partitioned and direct access files have default values assumed for them. VSAM files are not eligible and must be fully defined via JCL. No defaults are assumed for the INFILE data set.

Defaults are calculated to make sense, based upon the attributes of the original uncompressed data set. The File Prepass Utility saves information describing these attributes in the FDT, so they are available for use by the Compression and Expansion Utilities.

Extensive error checking is performed to ensure that the combination of user specifications and defaults assumed by the utilities (for both JCL and RDL) are compatible. When a JCL/RDL situation arises which makes sense, it is allowed to pass, often with a warning message. Only if it makes no sense at all does the utility end with an error message. Therefore, it is desirable to perform a trial compression and re-expansion of a few records to do the following:

- Check the defaults (JCL and RDL).

- Analyze possible warning messages.
- Check the RDL for “Record Definitions Imply Wrong Length”, etc.

See the Messages and Abend Codes chapter for a complete list of warning and error messages.

The DSORG Parameter

The DSORG parameter is only required for the OUTFILE data set if it is ISAM with DISP=NEW. If the data set has partitioned organization, a member name must be specified on the DSN parameter.

The RECFM Parameter

For the File Compression Utility, the default value for the compressed data set is RECFM=VB.

For the File Expansion Utility, the default value for the expanded data set is the same as the original uncompressed data set’s RECFM. Except for blocking and spanning, the RECFM (F, V, U) must be the same, even if coded by the user.

The LRECL Parameter

The LRECL of the compressed output from the File Compression Utility defaults to LRECL+8 of the original uncompressed data set. See Table 6-1 for guidelines when an explicitly coded LRECL is required. For output from the File Expansion Utility, the LRECL value assumed is that of the original uncompressed data set.

The BLKSIZE Parameter

For both the compressed output of the File Compression Utility and the expanded output of the File Expansion Utility, the value assumed for the BLKSIZE parameter depends upon the associated values for RECFM and LRECL, whether user-specified or assumed by default. For variable-length records written by the File Compression Utility, BLKSIZE is taken as the maximum of either the original uncompressed data set’s BLKSIZE or LRECL+4 of the compressed data set. For variable-length records written from the File Expansion Utility, BLKSIZE is taken as the maximum of either the BLKSIZE or LRECL+4 of the original uncompressed data set. For RECFM=F, FS or U, BLKSIZE is set equal to LRECL. For RECFM=FB or FBS, BLKSIZE is set equal to the highest integral multiple of LRECL which does not exceed the original uncompressed BLKSIZE. Neither utility checks to see if an assumed BLKSIZE is less than the track capacity. If the assumed BLKSIZE exceeds the track capacity, the user must explicitly code the BLKSIZE via JCL.

The KEYLEN Parameter

A default value for the KEYLEN parameter can be supplied. For both utilities, KEYLEN defaults to the original uncompressed data set’s KEYLEN value. It is assumed that the key is exempted from compression if the user has coded RDL specifi-

cations. A specification of KEYLEN=0 is treated the same as an omitted KEYLEN parameter.

The RKP Parameter

A default value for the RKP parameter can be supplied. For both utilities, in order to assume default RKP value, a default KEYLEN must be assumed. A specification of RKP=0 is treated the same as an omitted RKP parameter.

For the File Compression Utility, the compressed data set's RKP is set compatible with assumed or user-specified RDL specifications. For assumed RDL specifications, this procedure is automatic. For user-supplied RDL specifications, if RKP and KEYLEN are left to default, the key must be exempted from compression; i.e., contained within type N field definitions.

For the File Expansion Utility, the expanded data set's RKP is set equal to that of the original uncompressed data set.

The AMP Parameter

No default JCL parameters are assumed for VSAM files. The AMP parameter is specified as follows:

```
AMP=( AMORG, . . . )
```

Express File Expansion Utility

The Express File Expansion Utility expands a data set previously compressed by the Express Compression Utility. The entire data set is expanded. The expanded data set is identical to the data set as it existed prior to compression. Data integrity is ensured by comparison of the redundancy check byte appended to each compressed record when the data set was compressed, with a redundancy check byte calculated as each record is expanded.

The Express File Expansion Utility prints the INFILE and OUTFILE DCB information as in the File Compression Utility.

Execution Example—Express File Expansion Utility

The Express File Expansion Utility is invoked by executing the program named EXPANDX. Unlike the File Compression Utility, the Express File Expansion Utility does not offer the option to process only a portion of the input data set.

The OUTFILE DD statement defines the uncompressed data set written by the Express File Expansion Utility. The RECFM DCB attribute must be the same as the RECFM DCB attribute of the original uncompressed data set. So must the LRECL DCB attribute be the same; however, the number of records per block may differ. The values coded for primary and secondary space allocations can be those of the original uncompressed file provided that the file has not grown (through normal update processing) since it was compressed. Otherwise, the user must calculate ap-

appropriate values, based upon the original file's BLKSIZE, LRECL, number of records in the file being expanded, and the device type to which the expanded data set is written.

Figure 6-11 illustrates the expansion of the entire file, USER.FILE.

```

//*SHR2EXP3 JOB
//*
//*
//*****
//*
//*   THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS USERS TO   *
//*   PERFORM EXPANSION OF A SPECIFIED DATA SET                 *
//*
//*****
//EXPAND   EXEC PGM=EXPAND
//STEPLIB DD DISP=SHR,DSN=user.shrink.load
//PRINT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE  DD DISP=SHR,DSN=user.comprsd.file
//OUTFILE DD DISP(,CATLG),DSN=user.output.file,
//         DCB=(RECFM=recfm,BLKSIZE=blksize,LRECL=lrecl),
//         UNIT=unit,VOL=SER=volser,SPACE=(b,(p,s))

```

Figure 6-11. Express File Expansion Utility

Using SAMS:Compress/2 Subroutines

SAMS:Compress/2 provides the following five subroutines which can be invoked from users' application programs for processing of compressed data:

- SHRINK Converts an uncompressed record image to compressed form.
- EXPAND Converts a compressed record image to its original uncompressed form.
- PUFFUP From a compressed record, constructs a record image which contains all fields exempted from compression (i.e., type N fields) returned to their original displacements from the record image origin. The remainder of the constructed record image area is not modified.
- PUFFDOWN Reverses the process of the PUFFUP subroutine by moving fields exempted from compression from their original displacement in a "puffed-up" record image to the beginning of a compressed record image.
- CLOSE Frees storage dynamically acquired by calls to any of the above subroutines when it is no longer required.

These subroutines are fully re-entrant, and may reside in the Link Pack area, LINKLIB, or a load module PDS defined through STEPLIB or JOBLIB in the JCL.

Accessing the FDT

All SAMS:Compress/2 subroutines refer to the FDT associated with each compressed data set. The FDT for each compressed data set that is processed must be defined or made accessible via the JCL used to invoke the application program which uses SAMS:Compress/2 subroutines.

FDTs maintained in sequential data set format require separate definition on a TABLxx DD statement, where xx, coded as a two-digit number ranging from 00 through 31, corresponds to a file number supplied by the user as a parameter in the CALL to a SAMS:Compress/2 subroutine.

FDTs can be stored as members of a PDS in load module format. Coding the STEPLIB DD statement (or a concatenation with it) to define the PDS containing FDTs in load module format makes every FDT contained in the PDS accessible from the application program. In this manner, all FDTs that are used in an application program can be made accessible with a single JCL statement (provided they are all contained in the same PDS).

Within the application program, each FDT in load module format must have an associated 48-byte, fullword-aligned SAMS:Compress Control Block (SCB). Subroutine calls pass the SCB as a parameter to identify the FDT, instead of the file number as with FDTs in sequential data set format. Prior to the first subroutine CALL using a particular FDT, its associated SCB must be initialized as follows:

- The first eight bytes must contain the member name of the FDT in load module format, left-justified with trailing blanks as necessary to fill the full eight bytes.
- The remaining 40 bytes must contain binary zeros.

The following model definitions demonstrate properly initialized SCBs:

Assembler Language:

```
SCBNAME DS 0F
          DC CL8' fdt-member-name'
          DC 10F'0'
```

COBOL—must be in WORKING-STORAGE SECTION:

```
01  SCBNAME.
    03  FILLER PICTURE X(8) VALUE 'fdt-member-name'.
    03  FILLER PICTURE X(40) VALUE LOW-VALUES.
```

PL/I:

```
DCL 01  SCBNAME,
    03  name CHAR(8) INIT ('fdt-member-name'),
    03  N(10) FIXED BIN(31) INIT ((10)0);
```

After initialization, the user should not modify the SCB in any way. For re-entrant programs, the SCB must reside in dynamically allocated main storage.

JCL Implications for Existing Application Programs

JCL statements used to invoke existing application programs define uncompressed data sets. When the user incorporates calls to SAMS:Compress/2 subroutines into an application program to process compressed data sets, corresponding changes must be made to the JCL operands which invoke the application program. The definition of the compressed version of the data set must be substituted for the definition of each formerly uncompressed data set.

Information necessary to correctly code JCL parameters to define compressed data sets is available as a result of the job step which compressed the data set using the File Compression Utility. See the “Default JCL for the Compression and Expansion Utilities” section in this chapter for a discussion of commonly affected DCB parameters.

Calling the Subroutines

SAMS:Compress/2 subroutines are called when required by the application program using language-dependent conventions for subroutine linkage and parameter passing, as follows:

Assembler Language:

Normal OS calling conventions are observed. General registers 13, 14, 15 and 1 are used to contain the address of an 18-word save area, return address, subroutine entry address, and parameter list address, respectively. The parameter list contains the addresses of the parameters, and the end of the list must be marked. The CALL macro should be used with the VL option to create the parameter list and call subroutine.

COBOL:

Normal COBOL CALL statements are used. Parameters are data element names of the storage areas containing subroutine operands.

PL/I Optimizing Compiler:

The compiler must be informed that the SAMS:Compress/2 subroutines are Assembler subroutines by declaring all entry points, as follows:

```
DCL ( SHRINK ,EXPAND ,PUFFUP ,PUFFDOWN ,CLOSE ) OPTIONS ( ASM INTER ) ;
```

The subroutines can then be called using the standard PL/I CALL statement.

PL/I F Compiler:

There are special naming and parameter passing considerations for non-PL/I subroutines called from PL/I F programs. Once allowances have been made for the following considerations, the standard PL/I F CALL statement can be used to call SAMS:Compress/2 subroutines.

- External names must be 7 characters or less. PUFFDOWN must be given a 7-character name by using the linkage editor CHANGE statement when linking the subroutines with the application program.
- When passing strings or structures to the subroutines, the data address, not the dope vector, must be passed. This can be done by overlaying an arithmetic-based variable on the string or structure, and passing the arithmetic variable to the subroutine. See the IBM publication *PL/I F Programmer's Guide*, Chapter 15, "Communication with Other Languages."

For these languages, the application program performs all I/O. The first call to SHRINK, EXPAND, PUFFUP or PUFFDOWN referring to a particular file number or SCB causes the acquisition of 1K to 8K bytes of dynamic storage. The FDT is then read into storage, and specific code to perform the requested function is custom compiled. Subsequent calls execute the custom-compiled code only. The use of custom-compiled code provides the fastest possible execution time.

User Considerations

COBOL programs calling SHRINK subroutines cannot be compiled with the DYNAM option. Specifying the DYNAM option prevents the program from retrieving the desired SHRINK subroutine. Instead, the program retrieves the entry point of the corresponding SHRINK utility, because subroutines and utilities have like names.

General Sequence for Subroutine CALLS

The application program is responsible for performing all I/O. Once a compressed record is read, the application program normally does one of the following:

- CALL EXPAND and continue processing.
- CALL PUFFUP to determine, by examination of record keys, if the compressed record requires further processing. If it does not, the next compressed record is read, thus avoiding expansion overhead. If the compressed record requires processing, EXPAND can be called and processing can continue.
- CALL PUFFUP to determine if only "puffed up" fields are to be altered. If this is the case, processing continues, altering "puffed up" fields only. Thus, when processing for this record is complete, it is prepared to be written by calling PUFFDOWN. Otherwise, CALL EXPAND to make available the entire record contents, and continue processing.

Expanded records are processed by the application program as though they were never compressed. If a compressed data set is updated, record images must be compressed before they are written to the data set. Read-only data sets require no calls to SHRINK or PUFFDOWN. Similarly, unchanged records that are rewritten can avoid SHRINK (or PUFFDOWN) overhead by writing the original compressed record image. However, if the expanded record image is changed (or a new record image is to be

inserted), the application program must CALL SHRINK or CALL PUFFDOWN, as appropriate.

PUFFUP and PUFFDOWN incur far less processing overhead than SHRINK and EXPAND. This fact should be taken into consideration first when the RDL specifications are developed to compress the data set. Fields may be exempted from compression via RDL specification. Normally, these fields are record keys used for record retrieval, sort keys used by sort utility programs to reorder the file, or match keys used by application programs for file matching functions. In addition, the user developing RDL specifications should consider the trade-off between compression ratio and processing overhead involved in exempting a frequently processed non-key field from compression. Exempting one or more of these fields from compression is advantageous if the number of times the record must be expanded for further processing is significantly small, and the inclusion of the fields in uncompressed format can be accommodated without significantly reducing the compression ratio.

The CLOSE subroutine is used to free the storage obtained for the FDT and the custom-compiled compression/expansion code. In a batch application program which opens files, processes them, closes them, and goes to end-of-job, calling CLOSE is not necessary. In multistage batch application programs (which invoke a sort utility, for example) or in an online environment, calling CLOSE can be useful to assure maximum storage availability for subsequent processing stages.

Figure 6-12 illustrates the general sequence in which SAMS:Compress/2 subroutines are normally called.

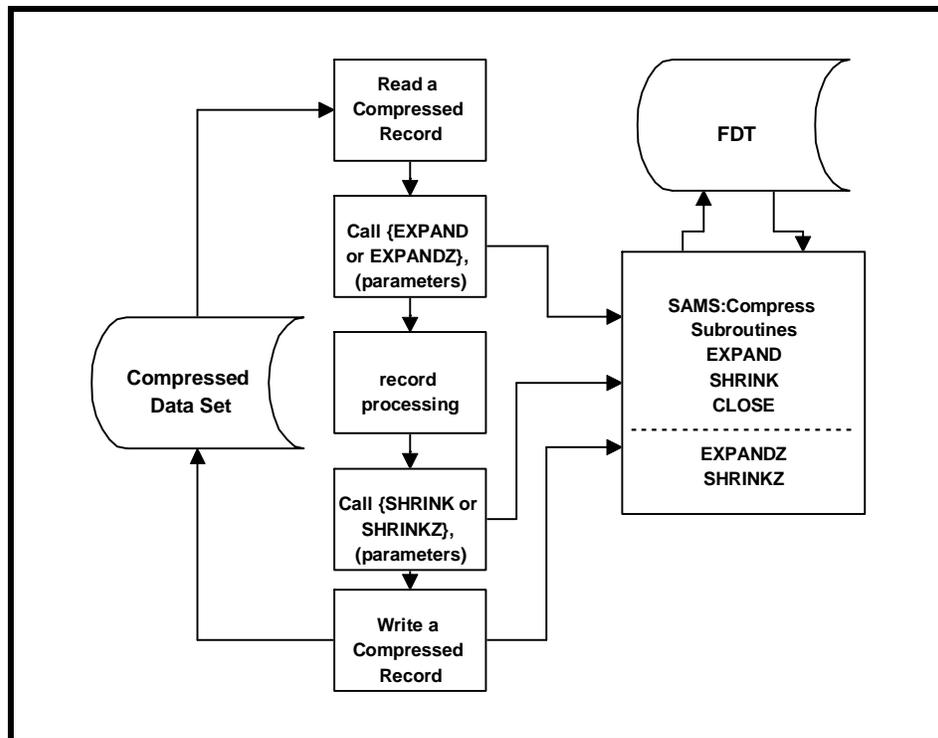


Figure 6-12. General Flow of Subroutines

CALL to Subroutine SHRINK

Each CALL to SHRINK compresses one record image in main storage. There are five parameters (four required and one optional) which the user codes in a SHRINK CALL. The parameters are positional; they must be coded in the same sequence as shown in Figure 6-12. The following are model statements for calling SHRINK:

Assembler Language:

```
CALL    SHRINK, (URA,CRA,URL,FDT[,RC]),VL
```

COBOL:

```
CALL    'SHRINK' USING URA CRA URL FDT [RC].
```

PL/I Optimizer:

```
CALL    SHRINK(URA CRA URL FDT [RC]);
```

Table 6-2. Parameters for Subroutine SHRINK

Parm	Contents
URA	Uncompressed Record Address (Required) The address of the in-core record to be compressed. The SAMS:Compress/2 subroutine does not alter this area. If the record is variable length, URA is the address of the RDW.
CRA	Compressed Record Address (Required) The address of a user-supplied main storage area at least 290 bytes larger than the uncompressed record. The SAMS:Compress/2 subroutine places the compressed record in this area in standard variable length format; i.e., preceded by the standard four-byte RDW.
URL	Uncompressed Record Length (Required) The address of a halfword in which the user supplies the length of the uncompressed record in binary form; i.e., USAGE COMPUTATIONAL for COBOL users. If variable-length records are compressed, URL should be the same as the URA parameter.
FDT	File Descriptor Table Identifier (Required) If the FDT is in sequential data set format, FDT is the address of a binary fullword which contains the file number (a value between zero and 31). This value must correspond to the value coded for xx in the TABLxx DD statement which defines the FDT. If the FDT is in load module format, the File Descriptor Table Identifier is the address of the SCB associated with this file (See "Accessing the FDT" above.)
RC	Return Code (Optional) The address of a binary fullword in which the SHRINK subroutine can store a value to indicate whether an error occurred in compressing the record. If this optional parameter is coded, certain ABEND conditions are suppressed, and a return code value is stored in the binary fullword addressed by RC, and in register 15 (the ANS COBOL special register for RETURN-CODE—see the Messages and Abend Codes chapter). If no error occurs during processing by the SHRINK subroutine, register 15 is zero upon return to the user's program, and RC is set to zero.

CALL to Subroutine EXPAND

Each call to EXPAND returns one compressed record image in main storage to its original uncompressed form. There are five parameters (four required and one optional) which the user codes in an EXPAND CALL. The parameters are positional; they must be coded in the same sequence as they appear in Figure 6-12. The following are model statements for calling EXPAND:

Assembler Language:

```
CALL EXPAND, (URA,CRA,URL,FDT,[,RC]),VL
```

COBOL:

```
CALL 'EXPAND' USING URA CRA URL FDT [RC].
```

PL/I Optimizer:

```
CALL EXPAND(URA CRA URL FDT [RC]);
```

Table 6-3. Parameters for Subroutine EXPAND

Parm	Contents
URA	Uncompressed Record Address (Required) The address of the main storage area where the EXPAND subroutine places the uncompressed record image which it constructs. If the record was a variable-length record before compression, URA is the address of the RDW of the expanded record image. The storage area provided by the user must be large enough to contain the entire expanded record.
CRA	Compressed Record Address (Required) The address of the compressed record image to be expanded by the subroutine. CRA must be the address of the data portion of the record, not the RDW. This storage area (i.e., the compressed record) is not modified by the EXPAND subroutine.
URL	Uncompressed Record Length (Required) The address of a halfword in which the subroutine returns the length of the expanded record in binary form; i.e., USAGE COMPUTATIONAL for COBOL users. If the record was a variable-length record before compression, URL can be identical to the URA parameter.
FDT	File Descriptor Table Identifier (Required) The address of a binary fullword which contains the file number (a value between zero and 31) if the FDT is in sequential data set format. If the FDT is in load module format, the File Descriptor Table Identifier is the address of the SCB associated with this file. (See "Accessing the FDT" above.)
RC	Return Code (Optional) The address of a binary fullword in which the EXPAND subroutine can store a value to indicate whether or not an error occurred in expanding the record. If this optional parameter is coded, ABEND 10 (check byte mismatch) is suppressed. Instead, a completion code of 4 is stored in the binary fullword addressed by the RC parameter, and also in register 15 (the ANS COBOL special register for RETURN-CODE). If no error occurs during EXPAND processing, register 15 is zero upon return to the user's program, and RC is set to zero.

CALL to Subroutine PUFFUP

Each call to PUFFUP produces a record image in which all fields exempted from compression (either by default or by user-specified record definitions) are located at the same location, relative to the beginning of the record image, as they were before compression. Figure 6-13 illustrates this process.

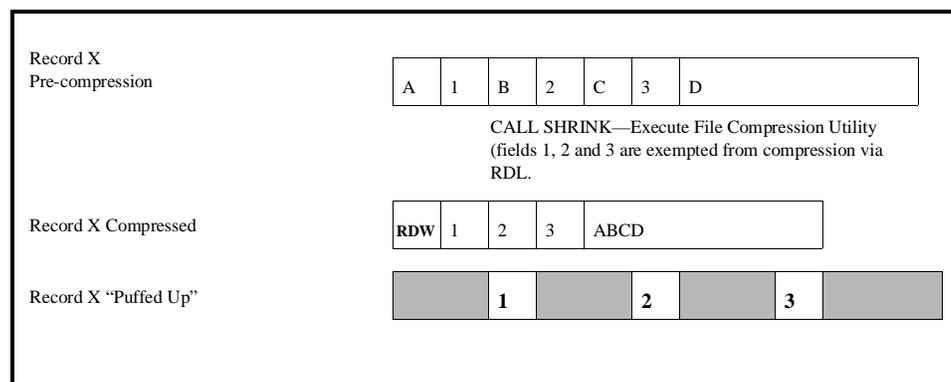


Figure 6-13. The PUFFUP Process

Areas of the “puffed up” record image that do not contain fields exempted from compression (the shaded areas of Record X “puffed- up” above) are not modified by the PUFFUP subroutine.

Processing overhead required to “puff up” a record is significantly less than that required to expand the record. By carefully selecting which fields to exempt from compression when a file is first compressed, unnecessary processing overhead in application programs which process the compressed file can be avoided. In effect, the technique is to make a PUFFUP call do the work of a call to EXPAND. This is particularly useful in applications which read a compressed file sequentially and perform detailed operations on a relatively small subset of the total file.

There are four parameters (all required) which the user codes in a PUFFUP call. The parameters are positional; they must be coded in the same sequence as they appear in Figure 6-12.

The following are model statements for calling PUFFUP:

Assembler Language:

```
CALL PUFFUP, (URA,CRA,RC,FDT),VL
```

COBOL:

```
CALL 'PUFFUP' USING URA CRA RC FDT.
```

PL/I Optimizer:

```
CALL PUFFUP(URA CRA RC FDT);
```

Table 6-4. Parameters for Subroutine PUFFUP

Parm	Contents
URA	Uncompressed Record Address (Required) The address of the main storage area where the PUFFUP subroutine places the “puffed up” record image it constructs. The storage area must be large enough to contain a complete uncompressed record, even though only fields exempted from compression are placed here.
CRA	Compressed Record Address (Required) The address of the compressed record image to be “puffed up” by the subroutine. CRA must be the address of the data portion of the record, not the RDW. The compressed record image is not modified by the subroutine.
RC	Return Code (Required) The address of a binary fullword in which the PUFFUP subroutine stores a value to indicate whether or not an error was detected during PUFFUP processing. A value of zero indicates no error was detected. A 4 indicates an invalid parameter list was passed. The return code value is also returned in register 15 (RETURN-CODE special register for ANS COBOL users).
FDT	File Descriptor Table Identifier (Required) The address of a binary fullword which contains the file number (a value between zero and 31) if the FDT is in sequential data set format. If the FDT is in load module format, the File Descriptor Table Identifier is the address of the SCB associated with this file.

CALL to Subroutine PUFFDOWN

Each call to PUFFDOWN reverses the process performed by the PUFFUP subroutine for one record image. All fields exempted from compression (either by default or by user-specified field type N record definitions) are taken from their normal locations within a “puffed up” record image and are returned to the compressed record image. Figure 6-14 illustrates this process.

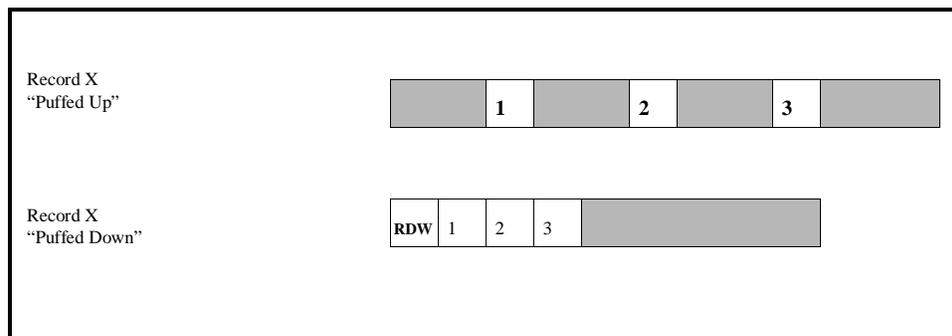


Figure 6-14. The PUFFDOWN Process

Areas of the “puffed up” and “puffed down” record images which do not contain fields exempted from compression (the shaded areas above) do not participate in the PUFFDOWN operation; they are not modified.

Following is the typical sequence of application program processing steps illustrating the use of a PUFFDOWN call:

1. Read a compressed record.
2. CALL PUFFUP.
3. Determine that only fields exempted from compression require further processing for the current record.
4. Process the current record, updating one or more fields exempted from compression.
5. CALL PUFFDOWN.
6. Write updated compressed record.

Step 3 decides if any compressed fields are required to complete processing for the current record by examining one or more fields exempted from compression. If one or more compressed fields are required, EXPAND must be called before processing further, and step 5 would be a call to the SHRINK subroutine.

Step 4 processes the “puffed up” or expanded record image. If no fields are modified, no further processing is necessary. If only fields exempted from compression are modified, calling PUFFDOWN is sufficient to prepare the record image to be written. If one or more compressed fields are updated, the SHRINK subroutine must be called before writing the record.

There are four parameters (all required) which the user codes in a PUFFDOWN call. The parameters are positional; they must be coded in the same sequence as they appear in Figure 6-12.

The following are model statements for calling PUFFDOWN:

Assembler Language:

```
CALL    PUFFDOWN, (URA, CRA, RC, FDT) ,VL
```

COBOL:

```
CALL    'PUFFDOWN' USING URA CRA RC FDT.
```

PL/I Optimizer:

```
CALL    PUFFDOWN (URA CRA RC FDT);
```

Table 6-5. Parameters for Subroutine PUFFDOWN

Parm	Contents
URA	Uncompressed Record Address (Required) The address of the main storage area which contains the “puffed up” record image to be “puffed down.” This area is not modified by the PUFFDOWN subroutine.
CRA	Compressed Record Address (Required) The address of the main storage area which contains the compressed record image. If the compressed record image contains an RDW, CRA is the address of the byte following the RDW. This compressed record image was previously “puffed up” to produce the record image addressed by the URA parameter, above. The PUFFDOWN subroutine replaces the fields exempted from compression in the compressed record image, with the fields exempted from compression from the “puffed up” record image.
RC	Return Code (Required) The address of a binary fullword in which the PUFFDOWN subroutine stores a value indicating whether or not an error was detected during PUFFDOWN processing. A value of 0 indicates no error was detected; a 4 indicates an invalid CALL. The return code value is also returned in register 15 (the ANS COBOL special register for RETURN-CODE).
FDT	File Descriptor Table Identifier (Required) The address of a binary fullword which contains the file number (a value between zero and 31) if the FDT is in sequential data set format. If the FDT is in load module format, the File Descriptor Table Identifier is the address of the SCB associated with this file.

CALL to Subroutine CLOSE

Each call to the CLOSE subroutine frees all main storage obtained by SAMS:Compress/2 in support of one compressed file. Application programs normally call CLOSE only if all processing is complete for the compressed file in question, and if more processing is done that might require the maximum available storage to continue.

SAMS:Compress/2 obtains main storage for each compressed file processed by an application program. The FDT, compression and expansion tables and custom-compiled code to perform subroutine functions occupy the storage obtained by SAMS:Compress/2 for each compressed file processed. The amount of storage obtained for a particular file varies, depending upon the extent and complexity of record definitions. It can vary from about 2K to as much as 24K per file, though approximately 6K is considered average.

Once CLOSE is called for a file, the FDT, compression and expansion tables and custom-compiled code to perform subroutine functions for the file are no longer available. Any subsequent call to a SAMS:Compress/2 subroutine referencing that file causes the FDT and compression and expansion tables to be brought into storage again, and custom code to be compiled again to support SAMS:Compress/2 subroutine functions for the file.

There is one parameter required in a call to CLOSE—the File Descriptor Table Identifier— which, if the FDT is in sequential data set format, is the address of a binary

fullword containing the file number (a value between zero and 31). If the FDT is in load module format, the File Descriptor Table Identifier is the address of the SCB associated with this file. (See “Accessing the FDT” above.)

The following are model statements for calling CLOSE:

Assembler Language:

```
CALL    CLOSE,(FDT),VL
```

COBOL:

```
CALL    'CLOSE' USING FDT.
```

PL/I Optimizer:

```
CALL    CLOSE(FDT);
```

Incorporating Subroutine Calls in Existing Application Programs

In most cases, incorporating SAMS:Compress/2 subroutine calls in an application program requires only minor changes. The JCL which invokes the application program contains DD statements to define data sets for processing. Users whose DD statements omit specification of parameters which are obtainable from the data set label need only make a few JCL changes, because the compressed data set label automatically provides correct values. Data set attributes explicitly specified on a DD statement or in a DCB macro, COBOL FD or PL/I file definition must be changed to reflect the compressed data set accurately.

During compression, the compressed record image may temporarily grow by up to 290 bytes greater than the length of the uncompressed record. Consequently, the area supplied to contain the compressed record must be 290 bytes longer than the uncompressed record. After compression is complete, the compressed record is almost always not greater than eight bytes longer than the uncompressed record, even in worst-case situations. (See Table 6-2.)

If input records are processed in locate mode (using a DSECT in Assembler Language or the FD DATA RECORD in COBOL), further adjustments are necessary. In Assembler Language, the general register used with the DSECT must be loaded with the address of the uncompressed record image produced by the EXPAND or PUFFUP subroutine before attempting to process the uncompressed record image. In COBOL, the entire uncompressed record definition should be moved from the FILE SECTION to the WORKING-STORAGE SECTION. It is then a simple matter to insert the proper call to EXPAND or PUFFUP immediately following the READ (or GET) for the compressed file. Few if any logic changes should be necessary following the subroutine CALL. Calls to SHRINK and PUFFDOWN can be handled in a similar manner.

Defining Compressed Records in COBOL Application Programs

In most cases, compressed records are variable-length records. This fact requires special attention in COBOL application programs.

Definition of variable-length records is accomplished via an

```
OCCURS DEPENDING ON data-name
```

clause, where the element defined by “data-name” contains a value which indicates the actual number of characters in the variable-length record. SAMS:Compress/2 generates this value and places it in the compressed record in response to a field type L record definition. (See Record Definition Language chapter.)

The following procedure is recommended for altering a COBOL application program to process a compressed file. This procedure is applicable only for files which contained fixed-length records prior to compression.

1. Use the field type L RDL specification as the first, or only, RDL specification when the file is compressed. This specification causes SAMS:Compress/2 to store the actual data length of each variable-length compressed record as a two-byte binary field at the beginning of each compressed record.
2. Move the definition(s) of the uncompressed data records from the FILE SECTION to the WORKING-STORAGE SECTION.
3. Change RECORDING MODE from F to V or S, as necessary.
4. Supply the following as the data record in the FILE SECTION for the compressed file (update the FD to refer to this record definition as DATA RECORD):

```
01 SHRUNK-RECORD.  
03 LENGTH          PICTURE 9(4) USAGE COMPUTATIONAL.  
03 SHRUNK-DATA PICTURE X OCCURS n TIMES DEPENDING ON LENGTH.
```

The user must substitute the maximum record length for the value n. The user may substitute other names for SHRUNK-RECORD, LENGTH, and SHRUNK-DATA, as appropriate.

5. Code a call to EXPAND or PUFFUP, as appropriate, and insert it immediately following the READ for the file in question. Code SHRUNK-RECORD (or any user-supplied name substituted in place of SHRUNK-RECORD) as the CRA parameter in the call to PUFFUP or EXPAND. Code the data record name of the record definition moved to the WORKING-STORAGE SECTION as the URA parameter in the call to PUFFUP or EXPAND.

6. If issuing calls to the SHRINK subroutine, provide the following compressed record area in the WORKING-STORAGE SECTION (not the FILE SECTION):

```

01 COMPRESS-AREA.
03 RDW PICTURE 9(5) USAGE COMPUTATIONAL.
03 SHRUNK-RECORD-OUT.
05 LENGTH-OUT PICTURE 9(4) USAGE COMPUTATIONAL.
05 SHRUNK-DATA-OUT PICTURE X OCCURS n TIMES
    DEPENDING ON LENGTH-OUT.
    
```

The user must substitute the value of the OUTFILE LRECL increased by 290 for the value n. The user may substitute other names for COMPRESS-AREA, RDW, SHRUNK-RECORD-OUT, LENGTH-OUT, and SHRUNK-DATA-OUT, as appropriate. Assuming names coded as shown, code COMPRESS-AREA as the second (CRA) parameter of the SHRINK subroutine call. If the output record definition in the FILE SECTION is coded:

```

01 UPDATED-SHRUNK-RECORD.
03 UPDATED-LENGTH PICTURE 9(4) USAGE COMPUTATIONAL.
03 UPDATED-SHRUNK-DATA PICTURE X OCCURS n TIMES
    DEPENDING ON UPDATED LENGTH.
    
```

the updated compressed record is written as follows:

```

MOVE LENGTH-OUT TO UPDATED-LENGTH.
WRITE UPDATED-SHRUNK-RECORD FROM SHRUNK-RECORD-OUT.
    
```

7. Compile using the NOTRUNC option.

Linking Subroutines With Application Programs

SAMS:Compress/2 provides two methods to link-edit the subroutines with application programs. In the first method, subroutines are link-edited with the user's application program by including the module SHRKEXPD via an explicit INCLUDE linkage editor control statement. This method results in making a copy of the subroutines a part of the application program load module.

The second method provides an improved capability for link-editing subroutines with the user's application program. SAMS:Compress/2 subroutines are link-edited by including the module SHRKSTUB via an explicit INCLUDE linkage editor control statement. This module contains entry points for all of the subroutines and functions as a loader. The first CALL to a particular subroutine causes SHRKSTUB to load the correct subroutine from the SAMS:Compress/2 library. Subsequent calls to that subroutine result in a direct branch to the previously loaded subroutine with several important advantages:

- By avoiding the proliferation of copies of SAMS:Compress/2 modules link-edited into user application programs, library space is conserved.

- Application programs do not have to be relink-edited when a new release of the SAMS:Compress/2 system is installed. The subroutine support can reside on a single shared library.
- Application programs link-edited with the SHRKEXPD module must be patched each time a Product Maintenance Request (PMR) fix is issued. Application programs link-edited with the SHRKSTUB module require no modification when a SM is issued; only the appropriate subroutine module on the SAMS:Compress/2 library need be fixed.
- The subroutines can be shared between tasks. Attached modules are link-edited only with the SHRKSTUB module (approximately 200 bytes).

To use this method, the SAMS:Compress/2 library must either be in LINKLIB or be defined as part of the STEPLIB or JOBLIB in the JCL which invokes application program execution. Figure 6-15 illustrates a model application program link-edit.

```
//LKED          EXEC PGM=IEWL,PARM='NCAL,LET....'  
//SHRNKLIB     DD  DSN=user.shrink.load,DISP=SHR  
.   
.   
.   
//SYSLIN      DD  *  
  INCLUDE     SYSLIB(user application program module)  
  INCLUDE     SHRNKLIB(SHRKSTUB)  
.   
.   
/*  
//
```

Figure 6-15. Application Link-Edit with Subroutines

Using SAMS:Compress/2 Under CICS

SAMS:Compress/2 provides data compression and expansion facilities to CICS/VS SAMS:Compress/2 users. The program operates under control of the IBM Operating Systems OS/VS1, OS/VS2 SVS or MVS.

This chapter describes the use of SAMS:Compress/2 in CICS application programs. It is directed toward the application programmer. For further information about using SAMS:Compress/2 with CICS application programs, refer to the material earlier in this chapter describing SAMS:Compress/2.

Install the Callable SHRINK Subroutines for CICS

SAMS:Compress for MVS, because it fully supports data sets within a CICS region, replaced these SHRINK subroutines. In the past, SAMS:Compress provided compression functions in CICS environments by calling these subroutines. Now, only the rarest circumstances justify their use. If you must use them, complete the steps in this section to call the SHRINK subroutines directly from programs running under CICS.

Callable SHRINK Subroutines for CICS Install Checklist

Complete the steps in this section to call the optional SHRINK subroutines from programs running under CICS

Complete	Activity
<input type="checkbox"/>	Load the CICS SAMS:Compress/2 source library data set from tape to disk. Use the JCL in INSTAL13.
<input type="checkbox"/>	Set the CWASHRK value in member SHRKWORD to point to a reserved fullword in the CWA. Change the member FDTNAMES to list the FDTs to be used.
<input type="checkbox"/>	Assemble and link-edit SAMS:Compress/2 source modules. Use the JCL MEMBER INSTALL in the CICS SAMS:Compress/2 source library.
<input type="checkbox"/>	Modify the CICS tables appropriately. Do both items below: <ul style="list-style-type: none"> • Define the SHRKNMOD and SHRKSCBS modules and all FDT modules to be used in the CICS Processing Program Table (PPT). • Define SHRKNMOD in the CICS Program List Table (PLT).
<input type="checkbox"/>	Modify the application programs calling SAMS:Compress/2 CICS subroutines and link-edit them with the SHRKCICS module.

Load the Source Modules

If you are installing SAMS:Compress to interface with CICS, the next job to run is SHRINK.SHR2.SOURCE(CICSLIB). If you are not going to interface SAMS:Compress with CICS, skip this section. CICSLIB loads the SAMS:Compress source modules from tape to disk.

Specify the FDT Names and a Fullword in the CWA

In the CICS SAMS:Compress/2 source library:

- Edit member SHRKWORD to specify the offset from the start of the CWA of an aligned fullword in the CWA to be used exclusively by SAMS:Compress.
- Edit member FDTNAMES to specify the names of the FDTs for the data sets used with SAMS:Compress.

If you need to change the offset of the reserved fullword, you must repeat this and the assembly and link steps, contained in INSTALL in the SAMS:Compress CICS source library. If converting from the macro level interface, you must change SHRKWORD to the offset from the CWA, *not* the CSA.

For seldom used files, you can omit the record in FDTNAMES specifying the corresponding FDT, because the FDT is loaded dynamically as needed, along with about 3K of storage for each transaction. This storage, and the dynamically loaded FDT, is freed when it is no longer needed. Since each FDT can use between 4 to 8K of storage, save storage in this way or, whenever appropriate, use a single FDT for several files.

CICS SAMS:Compress/2 can issue the following CICS transaction abends:

```
SHR1 - SHRKWORD NOT INITIALIZED OR DOES NOT POINT TO SHRKSCBS  
SHR2 - NO RETURN CODE PARAMETER PASSED TO SHRKSTUB
```

SHRNKMOD should be included in PLT, or you may run it as a startup transaction. CICS interface requires a return code argument which is optional in batch. This prevents CICS from ABENDING on SAMS:Compress/2 errors. Correct the application program so that it passes a return code parameter.

The names of FDTs prefixed by SCB in the SHRINK.SHR2.SOURCE(CICSLIB) member, are the FDTs that are loaded into storage at CICS startup.

Assemble and Link the Program Modules

Next, modify and run SHRINK.SHR2.SOURCE(INSTALL) to assemble and link the program modules. INSTALL contains two in-stream procedures.

- INIT deletes and reallocates the SHRINK CICS load library.
- CMDASM assembles and links the program modules.

Make the changes shown below to assemble and link the SHRNKMOD, SHRKCICS, and SHRKSCBS modules. If you need to change FDTNAMES, the member that holds the list of FDT names, you must assemble and link the SHRKSCBS module and update the PPT to reflect the current list of FDT names.

- The DSN of the Assembler H library.
- The DSN of the standard system macro library.
- The DSN assigned to SHRINK.SHR2.SOURCE. Refer to the LOADPDS step of the INSTAL13 job.
- The DSN of the CICS macro library.
- The DSN of the CICS source library.
- Link-edit parameters must default or be specified for “AMODE=24” and “RMODE=24”.
- The DSN assigned to SHRINK.LOAD. Refer to the LOADPDS step of the INSTALL1 job.

Satisfy CICS and Application Program Requirements

After you run CICSLIB, define SHRINK elements in CICS tables and link the application programs as follows:

- In the Processing Program Table (PPT), define SHRINKMOD, SHRKSCBS, and all FDT modules to be used via SAMS:Compress. Specify all PPT entries as PGMLANG=ASSEMBLER, PGMSTAT=ENABLED, and RELOAD=NO.
- In the Program List Table (PLT) that specifies programs to be executed during CICS initialization, define SHRINKMOD. Make sure that this PLT is identified by the PLTPI operand of the DFHSIT macro.
- If you are converting existing application programs from the macro-level product that was supplied with releases of SAMS:Compress before version 4.6.3, you must change the programs as described in the comments in member INSTALL in the SAMS:Compress CICS source library. In particular, the call to SHRKCICS must conform to command level specifications:
 - Register 13 must point to a valid save area
 - The 5 arguments required by SAMS:Compress must be preceded by the EIB and a dummy DFHCOMMAREA.
 - Link-edit the application programs with the SHRKCICS module for CICS. Do not use SHRKSTUB or SHRKEXPD.
 - Define the application programs in the PPT as usual.

SAMS:Compress/2 Subroutines Under CICS

SAMS:Compress/2 provides two subroutines for processing compressed data in CICS application programs. Each application program must be link-edited with the module SHRKCICS (rather than with SHRKSTUB or SHRKEXPD, as specified earlier in this chapter for SAMS:Compress/2). The subroutines are as follows:

SHRINK Converts an uncompressed record image to compressed form.

EXPAND Converts a record image to its original uncompressed form.

These differ slightly from the subroutines with these names in SAMS:Compress/2:

- The 5 arguments passed to the SHRINK or EXPAND subroutines must be preceded by EIB and DFHCOMMAREA, as required by command level CICS.
- The fourth parameter must be an SCB.

- The SCB must be 27 words long (rather than 12 words long).

In order to compress or expand a record, SAMS:Compress/2 must be given the name of its associated File Descriptor Table (FDT). The FDT name is passed through a user-defined SAMS:Compress Control Block (SCB). The SCB is a full-word-aligned 27-word area, the first 8 bytes of which contain the FDT name (left-justified and padded with blanks if necessary).

SCB initialization

Since the SAMS:Compress/2 subroutines modify the SCB, it must be in transaction-related dynamic storage. The following model statements demonstrate the SCB initialization:

COBOL Command Level—Must be in WORKING STORAGE SECTION

```
01 SCBNAME.  
   03 FILLER PICTURE X(8) VALUE 'fdtname'.  
   03 FILLER PICTURE X(40).
```

The CICS preprocessor converts the static WORKING-STORAGE SECTION to dynamic storage.

COBOL Macro Level

A structure equivalent to that given for COBOL Command Level must be created in storage dynamically acquired using CICS storage control facilities, but Macro Level should not be used with this Command Level interface.

PL/I

```
DCL 01 SCBNAME AUTOMATIC,  
     03 name CHAR(8) INIT ('fdt-member-name'),  
     03 N(10) FIXED BIN(31);
```

Assembler Language

A 12-word area either must be reserved in the transaction work area (TWA) or must be obtained dynamically by a DFHSC GETMAIN or the equivalent. If the SCB address is in register 1, it can be initialized by the instruction:

```
MVC 0(8,1),=CL8'fdt-member-name'
```

Calling Sequences for SAMS:Compress/2

The calling sequences for the three languages are as follows. The parameters of the calls are explained in Tables 6-6 and 6-7.

COBOL

```
CALL 'SHRINK' USING UR CR URL SCB RC.  
CALL 'EXPAND' USING UR CR URL SCB RC.
```

PL/I Optimizer

```

DCL      (SHRINK,EXPAND) OPTIONS (ASM INTER);
.
.
.
CALL SHRINK  (UR CR URL SCB RC);
CALL EXPAND  (UR CR URL SCB RC);

```

PL/I (F)

```

DCL  PRM1  BASED(P1)  FIXED  BIN(31);
DCL  PRM2  BASED(P2)  FIXED  BIN(31);
DCL  PRM3  BASED(P3)  FIXED  BIN(31);
DCL  PRM4  BASED(P4)  FIXED  BIN(31);
.
.
.
P1=ADDR(UR); P2=ADDR(CR);
P3=ADDR(URL); P4=ADDR(SCB);
.
.
.
CALL SHRINK  (PRM1,PRM2,PRM3,PRM4,RC);
CALL EXPAND  (PRM1,PRM2,PRM3,PRM4,RC);

```

Assembler Language

```

CALL SHRINK,(UR,CR,URL,SCB,RC),VL,MF=(E,SHRKLST)
CALL EXPAND ,(UR,CR,URL,SCB,RC),VL,MF=(E,SHRKLST)
.
.
.
* LAYOUT OF TASK-RELATED DYNAMIC STORAGE:
DSECT
.
.
.
SHRKLST  CALL  SHRINK,(UR,CR,URL,SCB,RC),MF=L
.
.
.

```

Note: The called subroutines from the application modify storage obtained by the start-up module, SHRKNMOD. Unless disabled, Intertest will issue a breakpoint on these instructions. Proceed and disregard the breakpoint. One such instruction that will cause a breakpoint is located at label SE in the CSECT SHRKCICS.

The tables below show the parameters for the SHRINK and EXPAND subroutines.

Table 6-6. Parameters for Subroutine SHRINK

Parm	Contents
UR	Uncompressed Record The in-storage record to be compressed, including the RDW, if present. The SHRINK subroutine does not alter this area.
CR	Compressed Record A user-supplied area at least 290 bytes larger than the uncompressed record. The SHRINK subroutine places the compressed record in this area in standard variable-length format; that is, starting with a standard four-byte RDW.
URL	Uncompressed Record Length A binary halfword (in COBOL, PIC S9(4) COMP) in which the user supplies the length of the uncompressed record in binary form. If the record has an RDW, URL can be the same as the UR parameter.
SCB	SAMS:Compress Control Block The SCB for this file.
RC	Return Code A binary fullword (in COBOL, PIC S9(5) COMP) in which the SHRINK subroutine may store a non-zero error code. If an error occurs, a return code value is stored in RC, and also in register 15 (in ANS COBOL, RETURN-CODE special register). If no error occurs, register 15 and RC are zero upon return.

Table 6-7. Parameters for Subroutine EXPAND

Parm	Contents
UR	Uncompressed Record An area where the EXPAND subroutine places the uncompressed record image it constructs, starting with an RDW if the original record had an RDW.
CR	Compressed Record The compressed record image to be expanded by the subroutine. CR must be the address of the data portion of the record, not the RDW, if any. This record image is not modified by the EXPAND subroutine.
URL	Uncompressed Record Length A binary halfword (in COBOL, PIC S9(4) COMP) in which the subroutine returns the length of the expanded record. If the record was a variable-length record before compression, URL can be identical to the UR parameter.
SCB	SAMS:Compress Control Block The SCB for this file.
RC	Return Code A binary fullword (in COBOL, PIC S9(5) COMP) in which the EXPAND subroutine may store a non-zero error code. If an error occurs, a return code value is stored in RC, and also in register 15 (in ANS COBOL, RETURN-CODE special register). If no error occurs, register 15 and RC are zero upon return.

VARISAM—Variable-length ISAM Subroutines

To achieve optimum compression, data sets compressed normally consist of standard form variable-length records (i.e., a four-byte Record Descriptor Word containing the length of the record) followed by the record. ANS COBOL does not support variable-length records for ISAM files. To enable COBOL users of ISAM files to process compressed ISAM data sets, SAMS:Compress/2 provides a set of subroutines, collectively termed VARISAM, to replace PROCEDURE DIVISION input/output statements which operate upon compressed ISAM data sets. Moreover, the subroutines are not restricted to COBOL but may be invoked from any language with standard CALL linkage.

The subroutines are VISOPEN, VISREAD, VISWRITE, VISREWRT, VISCLOSE and VISDELET. They replace the related COBOL OPEN, READ, WRITE, REWRITE and CLOSE statements. The VISDELET routine allows the deletion of records by key with minimal processing overhead. The subroutines cannot be used for fixed-length records.

Converting COBOL Programs to Use VARISAM Subroutines

Indexed sequential files are initially compressed using the File Compression Utility. All references to these files must be removed from the ENVIRONMENT DIVISION and FILE SECTION of the DATA DIVISION. Record descriptions must appear in the WORKING-STORAGE SECTION or LINKAGE SECTION. Statements in the PROCEDURE DIVISION referring to these files are changed to subroutine CALLs. Application program references in the PROCEDURE DIVISION to records or fields within records require no changes.

The generalized example shown in Figures 6-16 and 6-17 illustrates the process of converting a COBOL program to use VARISAM subroutines. Figure 6-16 presents a typical COBOL file processing structure, processing fixed-length ISAM files sequentially. Figure 6-17 presents that same COBOL program structure converted to use VARISAM to process compressed variable-length ISAM files.

```

IDENTIFICATION DIVISION.
.
.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
    SELECT INPUT-FILE ASSIGN TO DA-I-INPUT-FILE
        RECORD KEY IS INPUT-KEY.
    SELECT OUTPUT-FILE ASSIGN TO DA-I-OUTPUT-FILE
        RECORD KEY IS OUTPUT KEY.
DATA DIVISION.
FILE SECTION.
FD  INPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS F
    LABEL RECORD IS STANDARD
    DATA RECORD IS INPUT-RECORD.
01  INPUT-RECORD.
.
.   INPUT-KEY field is defined within the
.   INPUT-RECORD definition
FD  OUTPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS F
    LABEL RECORD IS STANDARD
    DATA RECORD IS OUTPUT-RECORD.
01  OUTPUT-RECORD.
.
.   OUTPUT-KEY field is defined within the
.   OUTPUT-RECORD definition
WORKING-STORAGE SECTION.
.
.
.
PROCEDURE DIVISION.
    OPEN INPUT INPUT-FILE, OUTPUT OUTPUT-FILE.
PROCESS-LOOP.
    READ INPUT-FILE AT END GO TO END-OF-JOB.
.
.
    WRITE OUTPUT-FILE.
    GO TO PROCESS-LOOP.
END-OF-JOB.
    CLOSE INPUT-FILE, OUTPUT-FILE.
    STOP RUN.

```

Figure 6-16. COBOL—Processing of Fixed-Length Uncompressed ISAM Files

```

IDENTIFICATION DIVISION.
.
.
.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 INPUT-WORD      PICTURE 9(5) USAGE COMP SYNC.
77 INPUT-DDN       PICTURE X(8) VALUE 'INPUTDD'.
77 INPUT-STATUS    PICTURE 9(5) USAGE COMP SYNC.
77 INPUT-IOTYPE    PICTURE X(5) VALUE 'INPUT'.
77 INPUT-ACSMETH   PICTURE X VALUE 'S'.
77 INPUT-FN        PICTURE 9(5) VALUE 0 USAGE COMP SYNC.
77 OUTPUT-WORD     PICTURE 9(5) USAGE COMP SYNC.
77 OUTPUT-DDN      PICTURE X(8) VALUE 'OUTPUTDD'.
77 OUTPUT-STATUS   PICTURE 9(5) USAGE COMP SYNC.
77 OUTPUT-IOTYPE   PICTURE X(6) VALUE 'OUTPUT'.
77 OUTPUT-ACSMETH  PICTURE X VALUE 'S'.
.
.
.
01 OUTPUT-SCB.
   03 OUTPUT-SCB-DDNAME PICTURE X(8) VALUE 'FDTMEMBR'.
   03 FILLER PICTURE X(40) VALUE LOW-VALUES.
01 INPUT-RECORD.
.
.
.
01 OUTPUT-RECORD.
.
.
.
PROCEDURE DIVISION.
   CALL 'VISOPEN' USING INPUT-WORD, INPUT-DDN, INPUT-STATUS,
       INPUT-IOTYPE, INPUT-ACSMETH, INPUT-FN.
   IF INPUT-STATUS IS NOT EQUAL TO ZERO, GO TO ERROR-ROUTINE.
   CALL 'VISOPEN' USING OUTPUT-WORD, OUTPUT-DDN, OUTPUT-STATUS,
       OUTPUT-IOTYPE, OUTPUT-ACSMETH, OUTPUT-SCB.
   IF OUTPUT-STATUS IS NOT EQUAL TO ZERO, GO TO ERROR-ROUTINE.
PROCESS-LOOP.
   CALL 'VISREAD' USING INPUT-WORD, INPUT-RECORD, INPUT-STATUS.
   IF INPUT-STATUS IS EQUAL TO 20, GO TO END-OF-JOB.
   IF INPUT-STATUS IS NOT EQUAL TO ZERO, GO TO ERROR-ROUTINE.
.
.
.
   CALL 'VISWRITE' USING OUTPUT-WORD, OUTPUT-RECORD, OUTPUT-STATUS.
   IF OUTPUT-STATUS IS NOT EQUAL TO ZERO, GO TO ERROR-ROUTINE.
   GO TO PROCESS-LOOP.
ERROR-ROUTINE.
.
.
.
END-OF-JOB.
   CALL 'VISCLOSE' USING INPUT-WORD.
   CALL 'VISCLOSE' USING OUTPUT-WORD.
   STOP RUN.

```

Figure 6-17. COBOL—Converted to Use VARISAM to Process Compressed Variable-Length ISAM Files

Note: In Figure 6-17, the sample COBOL program structure is converted from the COBOL program structure illustrated in Figure 6-16. All references to the ISAM files have been removed from the ENVIRONMENT DIVISION and the FILE SECTION of the DATA DIVISION. Also, note that the record definitions have been moved from the FILE SECTION to the WORKING-STORAGE SECTION.

VARISAM Subroutine Descriptions and Coding Instructions

The following subsections describe each individual VARISAM subroutine and provide coding instructions for each.

VISOPEN Subroutine:

The VISOPEN subroutine initiates the processing of an input, output, or input-output file. VISOPEN determines whether file processing is random (R) or sequential (S). A file may only be created sequentially. VISOPEN must have successfully completed before any other VARISAM subroutines can be called for this file. The calling sequence for the VISOPEN subroutine is:

```
CALL 'VISOPEN' USING WORD, DDN, STAT, IOTYPE, ACSMETH [FN].
```

where:

WORD is an aligned fullword which is set by VISOPEN. WORD is passed, unchanged, in each successive VARISAM CALL which applies to this file. The user must not modify this parameter in any way. The following is a sample WORKING-STORAGE SECTION definition for this parameter:

```
77 WORD PICTURE 9(5) USAGE COMPUTATIONAL SYNC.
```

DDN is an eight-byte character string specifying the DDNAME (from the DD statement) of the file that is opened. The following is a sample WORKING-STORAGE SECTION definition for this parameter:

```
77 DDN PICTURE X(8) VALUE 'ddname'.
```

STAT is a binary fullword in which VISOPEN returns a completion code. The same code value is returned in register 15, the ANS COBOL RETURN-CODE special register. The following is a sample WORKING-STORAGE SECTION definition for this parameter:

```
77 STAT PICTURE 9(5) USAGE COMPUTATIONAL SYNC.
```

The meaning of the returned values are:

- 0 = normal completion
- 4 = invalid calling sequence
- 8 = could not open file (no DD statement, etc.)
- 12 = DCB RECFM not variable

IOTYPE is an alphabetic character string specifying “INPUT”, “OUTPUT” or “I-O”, depending on file usage. Files that are created must be specified as “OUTPUT”; sequential files that are updated are “I-O”, and random files that are updated are either “OUTPUT” or “I-O”. The following are sample WORKING-STORAGE SECTION definitions for this parameter:

```
77 IOTYPE PICTURE X(5) VALUE 'INPUT'.
77 IOTYPE PICTURE X(6) VALUE 'OUTPUT'.
77 IOTYPE PICTURE X(3) VALUE 'I-O'.
```

ACSMETH is a one-character string for sequential (QISAM) or random (BISAM) processing. The following are sample WORKING-STORAGE SECTION definitions for this parameter:

```
77 ACSMETH PICTURE X VALUE 'S'.
77 ACSMETH PICTURE X VALUE 'R'.
```

FN is an optional aligned binary fullword parameter containing a file number, or an optional parameter indicating a SAMS:Compress Control Block (SCB). If this parameter is coded, subsequent calls to VISREAD for this file automatically expand the input record before presenting it to the application program, and subsequent calls to VISWRITE or VISREWRT automatically compress the output record before writing it to the output data set. If this parameter is omitted, no automatic record expansion or compression is performed. The application program is presented a compressed record from VISREAD and is responsible for presenting a compressed record to VISWRITE or VISREWRT.

If the FDT associated with the compressed data set is in sequential data set format, FN is coded as a binary fullword containing the file number. The file number is a value of zero through 31, and must correspond with xx of a TABLxx DD statement submitted with the application program’s execution JCL to define the FDT.

If the FDT associated with the compressed data set is in load module format, FN specifies an SCB, coded in the WORKING-STORAGE SECTION. The SCB specifies the member name of the FDT in load module format. The partitioned data set containing the FDT in load module format must be specified on the STEPLIB DD statement (or concatenated to the STEPLIB DD statement) in the application program’s execution JCL.

The sample COBOL program in Figure 6-17 shows an example of each way to code this parameter. The INPUT-FILE is coded with a file number (in this case, zero); the OUTPUT-FILE is coded with an SCB.

If, for an appreciable number of records, only type N fields (i.e., fields exempted from compression) are inspected and/or modified by the application program, there

is no need to go through the unnecessary processing overhead inherent in compressing and expanding every record. In this case, it is recommended that this parameter not be coded. Instead, the user can call the PUFFUP subroutine, and only call the EXPAND subroutine for records with compressed fields that require processing.

VISREAD Subroutine

The VISREAD subroutine is used to read a record sequentially or by key into a user-supplied record area. If the FN parameter is coded for the VISOPEN subroutine for this file, the record is first expanded to its original uncompressed form. Variable-length records, whether compressed or uncompressed, are passed to the caller, including the four-byte RDW (record descriptor word). The calling sequence for the VISREAD subroutine is:

```
CALL 'VISREAD' USING WORD, RCD-AREA, STAT [,KEY]
```

where:

- WORD** is the fullword specified in the VISOPEN CALL for this file.
- RCD-AREA** is an area into which the input record is moved (with RDW if variable-length). In COBOL, the record definition must appear in the WORKING-STORAGE or LINKAGE SECTION. If the FN parameter is coded for the VISOPEN subroutine CALL for this file, the record supplied is already expanded. Otherwise, the record supplied is the compressed record read from the data set.
- STAT** is a fullword binary completion code containing the same value as the ANS COBOL RETURN-CODE special register (15). Completion code values supplied can be one of the following:
- 0 = normal completion
 - 4 = invalid calling sequence
 - 8 = file not open (bad return code from VISOPEN)
 - 12 = I/O error
 - 16 = SHRINK error
 - 20 = end-of-file if a sequential read; invalid if a keyed read
 - 24 = other error
- KEY** is a character string parameter containing the key of the record that is read. KEY is required if the file is read randomly; i.e., the value supplied for the ACSMETH parameter on the VISOPEN CALL for this file is R. KEY is optional if the file is read sequentially; i.e., the value supplied for the ACSMETH parameter on the VISOPEN CALL for this file is S. For sequential processing, coding the KEY parameter sets the position within the file where sequential processing continues. In this case, the application program must preset the value of the STATUS parameter to one of the values below. See the IBM manual *Data Management Services* for more information.

0 = read the record with the given key (SETL K)
 1 = read first record with key equal or greater (SETL KH)
 2 = read the first record in the given key class (SETL KC)

Omitting the KEY parameter for a sequential read indicates that the next (or first) record is read. If an attempt is made to issue a non-keyed sequential read beyond an EOF condition (STAT=20), unpredictable results occur.

VISWRITE Subroutine

The VISWRITE subroutine writes a record supplied by the application program either sequentially or by key, depending upon the value supplied for the ACSMETH parameter on the VISOPEN subroutine CALL for this file. If the ACSMETH value supplied is S, a sequential write (i.e., a PUT) is performed. If the ACSMETH value supplied is R, a keyed write is performed. The key is obtained from the record. The position and length of the key within the record is determined from the RKP and KEYLEN DCB attributes of the file.

If the FN parameter is coded on the VISOPEN subroutine CALL for this file, the user-supplied record is automatically compressed by the VISWRITE subroutine before it is written. Variable-length records must be presented with the four-byte RDW. The calling sequence for the VISWRITE subroutine is:

```
CALL 'VISWRITE' USING WORD, RCD-AREA, STAT.
```

where:

WORD is the fullword specified in the VISOPEN CALL for this file.

RCD-AREA is the record that is (optionally compressed and) written. If the FN parameter is coded for the VISOPEN subroutine CALL for this file, the record in this area should be in uncompressed format. VISWRITE compresses the record before writing it to the data set. Otherwise, the application program must ensure that a compressed record is presented to VISWRITE.

STAT is a fullword binary completion code containing the same value as the ANS COBOL RETURN-CODE special register (15). Completion code values supplied can be one of the following:

0 = normal completion
 4 = invalid calling sequence
 8 = file not open (bad return-code from VISOPEN)
 12 = I/O error
 16 = SHRINK error
 20 = invalid embedded key
 24 = other error

If the ACSMETH parameter value supplied on the VISOPEN subroutine CALL for this file is R (random), the VISWRITE subroutine can be used for the rewrite function. A record whose key already exists on the file can be written. The existing record is replaced, and a status code of 0 is returned. To emulate the INVALID KEY clause of COBOL, the VISREAD subroutine must be called before calling VISWRITE. This tests the invalid key condition (status code 20 from VISREAD).

VISREWRT Subroutine

If the previous CALL was a successful VISREAD and the key of the record was not modified, then the VISREWRT subroutine is applicable and more efficient than VISWRITE. This routine is especially useful in deleting records. (See DCB option OPTCD=L in the IBM Data Management Services manual.)

If the FN parameter is coded on the VISOPEN subroutine CALL for this file, the user-supplied record is automatically compressed by the VISREWRT subroutine before it is written. The calling sequence for the VISREWRT subroutine is:

```
CALL 'VISREWRT' USING WORD, RCD-AREA, STAT.
```

where:

- | | |
|-----------------|---|
| <i>WORD</i> | is the fullword specified in the VISOPEN CALL for this file. |
| <i>RCD-AREA</i> | is the record that is (optionally compressed and) rewritten. If the FN parameter is coded for the VISOPEN subroutine CALL for this file, the record in this area is in uncompressed format. VISREWRT compresses the record before writing it to the data set. Otherwise, the application program must ensure that a compressed record is presented to VISREWRT. |
| <i>STAT</i> | is a fullword binary completion code containing the same value as the ANS COBOL RETURN-CODE special register (15). Completion code values supplied can be one of the following: |
| | 0 = normal completion |
| | 4 = invalid calling sequence |
| | 8 = file not open (bad return code from VISOPEN) |
| | 12 = I/O error |
| | 16 = SHRINK error |
| | 20 = invalid key or wrong length record |
| | 24 = other error |

VISCLOSE Subroutine

This subroutine is called when processing of a data set is complete. It closes the data set, frees dynamic storage obtained by VARISAM and, if necessary, calls the SAMS:Compress/2 CLOSE subroutine. Subsequent VISCLOSE CALLs for the same file have no effect. The calling sequence for the VISCLOSE subroutine is:

```
CALL 'VISCLOSE' USING WORD.
```

where:

WORD is the fullword specified in the VISOPEN CALL for this file.

VISDELET Subroutine

This subroutine is used to delete a record whose key is known. It can only be used under the following conditions:

- The DCB parameter RKP must be greater than 4.
- If a compressed file is used, the first byte of the first type N field must not be part of the ISAM key.
- The DCB parameter OPTCD=L is specified in the JCL for this file.

The data set can be either sequential (S) or random (R). If the keyed record is not on the file, VISDELET does nothing. There is no SHRINK/EXPAND overhead involved. The calling sequence for the VISDELET subroutine is:

```
CALL 'VISDELET' USING WORD, KEY, STAT.
```

where:

WORD is the fullword specified in the VISOPEN CALL for this file.

KEY is a character string parameter containing the full key of the record that is to be deleted.

STAT is a fullword binary completion code containing the same value as the ANS COBOL RETURN-CODE special register (15). Completion code values supplied can be one of the following:

- 0 = normal completion
- 4 = invalid calling sequence
- 8 = file not open (bad return code from VISOPEN)
- 12 = I/O error
- 16 = other error

Link-Editing VARISAM Subroutines With User Application Programs

SAMS:Compress/2 provides two methods to link-edit VARISAM subroutines with applications programs. In the first method, both SAMS:Compress/2 subroutines and VARISAM must be link-edited with each application program that uses

VARISAM. Figure 6-18 illustrates this procedure. However, if this procedure is employed, each application program must be patched whenever a Product Maintenance Request (PMR) fix is issued.

```
//LKED      EXEC  PGM=IEWL,PARM='link-edit parameters'  
//SHRNKLIB DD DSN=user.shrink.load,DISP=SHR  
.standard link-edit DD statements  
.  
//SYSLIN   DD   *  
INCLUDE   SYSLIB(user application program)  
INCLUDE   SHRNKLIB(SHRKEXPD,VARISAM)  
.  
.  
/*  
//
```

Figure 6-18. VARISAM Link-Edit to Application—For Version Before 3.0

To avoid the difficulty of patching, SAMS:Compress/2 also provides an alternate procedure. Both SAMS:Compress/2 subroutines and VARISAM subroutines are link-edited with the user's application program by including the module SHRKSTUB via an explicit INCLUDE linkage editor control statement. This short module contains entry points for all SAMS:Compress/2 subroutines (SHRINK, EXPAND, PUFFUP, PUFFDOWN, and CLOSE) and VARISAM subroutines (VISOPEN, VISREAD, VISWRITE, VISREWRT, VISCLOSE, and VISDELETE) and functions as a loader. The first call to a particular subroutine causes SHRKSTUB to load the correct subroutine from the SAMS:Compress/2 library. Subsequent calls to that subroutine result in a direct branch to the previously loaded subroutine. This offers several important advantages:

- By avoiding link-editing of SAMS:Compress/2 subroutine modules with each user application program, library space is conserved.
- Application programs do not have to be relink-edited when a new release of the SAMS:Compress/2 system is installed. The subroutine support can reside on a single shared library.
- Application programs link-edited with the SHRKSTUB module require no modification when an SM is issued; only the appropriate subroutine module on the SAMS:Compress/2 library need be fixed.
- The subroutines can be shared between tasks. Attached modules are link-edited only with the short SHRKSTUB module (approximately 200 bytes).

In order to take advantage of this method, the SAMS:Compress/2 library must be defined as part of the STEPLIB or JOBLIB in the JCL which invokes the application program.

Figure 6-19 illustrates this recommended approach to link-editing application programs with VARISAM.

```
//LKED      EXEC  PGM=IEWL,PARM='link-edit parameters'  
//SHRNKLIB DD  DSN=user.shrink.load,DISP=SHR  
.standard link-edit DD statements  
.  
//SYSLIN   DD  *  
           INCLUDE  SYSLIB(user application program)  
           INCLUDE  SHRNKLIB(SHRKSTUB)  
.  
.  
/*  
//
```

Figure 6-19. VARISAM Link-Edit with User Application

Chapter 7. Record Definition Language

Overview

With the SAMS:Compress Record Definition Language (RDL), a user may describe characteristics of data comprising a file that is compressed (and expanded) by the SAMS:Compress system. The more the user knows about the data, the more detailed and precise the RDL specifications can be.

SAMS:Compress assumes default record definitions (described at the end of this chapter) if the user chooses not to code RDL specifications. The effectiveness in achieving impressive compression ratios using default record definitions (and corresponding low processing overhead) is considerable. In the absence of other considerations (e.g., the need to exempt specific non-key fields from compression, for which user-coded RDL specifications are necessary) it is recommended that the default definitions assumed by the system be used in first attempts to compress any file. Then, if performance is satisfactory, no further user coding is necessary.

On the other hand, a great deal of time and effort has been expended to ensure that SAMS:Compress's implementation of its various compression/expansion algorithms consumes the least possible number of CPU cycles. Users interested in achieving the best possible trade-off between compression ratio and processing overhead for a particular file are encouraged to experiment with alternative RDL specifications. An indication of the potential impact of coding a particular RDL specification on this trade-off can be found under individual RDL specification descriptions contained herein. This chapter covers the following topics:

- Performance Considerations When Using RDL
- How the RDL Operates
- RDL Terminology
- RDL Syntax
- RDL Field Type Descriptions
- RDL Repetition and Condition Groups
- RDL Position Function
- General Restrictions on RDL Use
- Guide to Correct RDL Specification
- RDL Defaults

Performance Considerations When Using RDL

There are two primary performance characteristics of the SAMS:Compress data compression system:

Compression Ratio	The amount of storage space saved by compressing a data set.
Processing Overhead	The amount of additional CPU cycles required to transform record images from the compressed state to the uncompressed state for processing by an application program, and to retransform records to the compressed state for storage in the data set.

Maximizing the compression ratio represents a cost saving for users, because more data can be stored per unit of storage available. The accompanying increase in processing overhead may or may not represent an increased cost to the user, depending upon the circumstances unique to each user.

Many factors influence whether the increase in processing overhead results in a cost or a saving to the user. Although it seems that increased processing overhead to compress and expand records always costs the user more, more records can be stored per block by compressing records, reducing the number of times that the IBM Operating System File Access Method must be entered per program execution. This reduction represents a decrease in processing overhead and may result in a net saving. The availability of CPU cycles during a typical job mix is also important. The total number of CPU cycles available per unit of time is a fixed cost to the user, directly related to the computing power of the user's installed hardware. If there are CPU cycles available during a typical job mix, increased processing overhead may result in no increased cost.

In any event, there is a trade-off relationship between compression ratio and processing overhead. As compression ratio approaches the theoretical maximum (i.e., storing the greatest amount of data in the fewest number of bits), processing overhead tends to increase. By benchmark testing, users may determine the optimum trade-off for themselves based upon their requirements.

How the RDL Operates

The theoretical maximum compression ratio is different for each data set, because it depends upon the actual data contained in the data set. The SAMS:Compress system provides multiple algorithms for compressing data, which the user can selectively apply to individual fields within data records to maximize their compression.

This compression is accomplished by using RDL specifications, which the user codes and supplies as input to build the File Descriptor Table (FDT). The more completely and accurately the user describes the records that are compressed

through RDL specifications, the closer the compression ratio approaches the theoretical maximum. However, each RDL specification coded by the user has an associated cost in processing overhead.

For example, a user knows that a certain field contains textual data, such as a customer name. It may also be known that the customer name field never contains numeric characters. This characteristic can be used to differentiate this field from a customer address field which, while containing textual information, does contain numeric characters. By using two different RDL specifications to define these fields, the user can achieve a higher compression ratio for both fields than if both fields are defined by the same RDL specification.

While it is often sufficient to know what kind of data is in a field, it is also helpful to know the distribution of values contained in the field across the file. For example, one of the most efficient ways to define a field to SAMS:Compress is as a small set of fixed expected values. A file may contain a warehouse name field, where there is only a small number of warehouses (assume eight) represented on the file. An RDL specification can be coded which provides these names as a set of expected values; i.e., a table.

RDL Terminology

To understand how to use RDL, certain technical terms used to describe the RDL in the following discussion must be understood. RDL specifications are coded by the user to describe the records which comprise the file that is compressed and expanded. Each RDL specification defines one data field.

A field, to SAMS:Compress, is a series of consecutive byte locations, the contents of which have similar compression/expansion characteristics, as determined by the user. The boundaries of fields defined to SAMS:Compress need not correspond with actual field boundaries of data elements. For example, the entire record can be defined to SAMS:Compress as a single SAMS:Compress field. Unless otherwise noted, future references to the term “field” in this chapter denote a field as defined to SAMS:Compress, not a record data element.

Fields are differentiated by the type of data they contain; e.g., character data, packed decimal data, etc. Consequently, each RDL specification consists, in part, of a field type code. From the field type code specified by the user, the SAMS:Compress system selects a compression/expansion algorithm appropriate for the field’s content.

In addition to the type of data contained in a field, SAMS:Compress must know the extent of the field; i.e., where it begins and ends—the boundaries of the field. Thus, each RDL specification coded by the user contains an indication of the length of the field, in bytes.

The location in the record where the field begins is implied by the sum of the lengths of fields previously defined. SAMS:Compress evaluates user-coded RDL

specifications left-to-right and maintains an internal field pointer (IFP). The value in the IFP is initially zero, corresponding to the first position of the record being defined. SAMS:Compress automatically adjusts the IFP value for each RDL specification, increasing it by the length of the previous field definition.

In a few special cases, the user may need to set the value of the IFP explicitly by coding a special RDL specification, the Position Function. Use of the Position Function is described later in this chapter.

Field lengths are not always fixed. The field length of a variable-length field must be determinable from information contained in the record. A separate field normally contains the length of a variable-length field, or contains a value indicating the number of times that a variably occurring fixed-length segment appears. SAMS:Compress allows users to perform arithmetic within a field definition to calculate a variable field length, using a special symbol, VS, to represent the calculated value. The VS (Variable Symbol) can be coded in certain field definitions in lieu of an integer length. Detailed description of calculating a value for the VS is presented later in this chapter.

The user may need to repeat an RDL specification two or more times in succession. For example, the record might contain 20 successive packed decimal numbers of identical length. To reduce user coding in such cases, the RDL provides for specification of a repetition factor for a single RDL specification, or a group of RDL specifications coded consecutively. This RDL specification structure has the same effect as coding the RDL specification or a group of RDL specifications as many times, in sequence, as indicated by the repetition factor. This RDL specification structure is referred to as a repetition group. The VS can be coded in cases where the repetition factor is variable and can be calculated from information contained in the record. This process is described later in this chapter.

Files may contain multiple record formats, where the format of a particular record can be determined from the contents of one (or more than one) individual field. SAMS:Compress allows alternative record definitions, which are effective for particular records based upon the contents of a field. Such an RDL coding structure is referred to as a condition group, and is described later in this chapter.

RDL Syntax

Syntax rules for the Record Definition Language are:

- Definitions appear within Columns 1-72 of each card.
- Definitions can be continued onto any number of cards.
- Each field definition consists of a one- or two-character type code followed by a field length descriptor.
- Definitions are separated by commas and/or blanks.

- Groups of definitions may be enclosed within quotes and preceded by a repetition factor.
- Condition groups (definitions whose pertinence is dependent on record content) are enclosed in parentheses.
- Numbers appearing in the language, either as field lengths or arithmetic constants, must be between 1 and 32767 (inclusive), unless otherwise specified.
- Definitions are terminated with a period or by end-of-file on the RECDEF data set. Information appearing after the period is treated as a comment.

As the record definitions are processed by the File Prepass Utility or the Interactive Dialog, they are checked for syntactical validity (but not applicability to the data) and printed. If syntax errors are encountered, each error is underscored by an alphanumeric character identifier, which corresponds to the initial character of an explanatory error message printed directly below the RDL statement in error.

Each RDL specification consists of a field type code and a field length descriptor. Valid field type codes are listed in Table 7-1.

Table 7-2 presents valid forms for coding field length descriptors.

Table 7-1. SAMS:Compress Field Types

Field Type	Type of Data Defined
C1, C2, C3	Character data using internal frequency table 1, 2, or 3, as specified
GA	Garbage, filler, padding, alignment bytes, etc.
L	Insert binary length indication (for COBOL users)
MA, MB	Pattern matching
N	Exempt from compression (keys, etc.)
PD	Packed decimal data
S, X	Set of expected values
UN	Undefined field
V, VP, VZ	Variable definition
ZL, ZR	Zoned decimal, left- or right-justified, as specified

Table 7-2. Field Length Descriptors

Code	Description
Fn	Fixed-length field of length n, where 1<n<16384; n may have leading zeros but may not exceed 8 decimal digits.
FVS	Length determined by the previous type-V field. This descriptor is valid only for types C1, C2, C3, UN, and GA.
VER	Variable-length field extending to the end of the record. This descriptor is valid only for types C1, C2, C3, UN, and GA.
Dc	Variable-length field delimited by an EBCDIC character, "c", or end of input record, whichever comes first. The length must be less than 128 bytes. This descriptor is valid only for types C1, C2, C3, UN, and GA. The field definition, if any, begins beyond the delimiter.

Below is an example of the use of field definitions with length specifications where:

- The first eight-byte field is a right-justified, zoned decimal number.
- The next 100-byte field is character data.
- The next two fields are four-byte packed decimal numbers.
- The remainder of the record is treated as character data.

ZRF8, C1F100, PDF4, PDF4, C1VER.

RDL Field Type Descriptions

The following sections describe individual RDL field type codes, with suggestions and restrictions concerning their use.

Field Types C1, C2, and C3—Character Data

These field types are compressed using the Huffman algorithm, coupled with elimination of successive repetitions of the same byte value. The value in each byte is assigned a variable-length bit code, with the most-frequently occurring value assigned the shortest bit code and the least-frequently occurring value assigned the longest bit code. The frequency of occurrence of each value is determined during the prepass and is stored in one of three character frequency tables. A separate character frequency table is associated with each of the character-type RDL field specifications C1, C2 and C3. When coding RDL specifications for type C fields, the user should attempt, to the greatest degree possible, to group together in the same frequency table those fields whose byte values are likely to have a similar distribution.

For example, the user could define predominantly alphabetic fields as type C1, predominantly numeric fields as C2, and fields with another kind of distribution as C3. The compression ratio thus obtained will be better, at no increase in processing overhead, than if all fields are defined as the same type. With the exception that

types C1, C2 and C3 have their own individual frequency table, they are treated identically by SAMS:Compress.

For example, on a name and address file, suppose the name appears in the first 40 positions, the street address in the next 39, the city and state in the next 28, and the ZIP code in the final six positions. The code could be:

C1F113.

but

C1F40, C2F39, C3F28, C2F6.

gives better results. After the prepass, the C1 table is heavily skewed toward alphabets, with the letters M, R, S, blank and the vowels used most frequently. Note that the more frequently the character is used, the shorter is its bit code representation. The second field is preponderantly alphabets, numerics, and blank so that its compression can be improved using a separate frequency table, C2. Since the last field is numeric, it can also be grouped in C2, although it probably is better for both fields to code it ZRF6. The city and state field may have the same approximate distribution as the first C1 field, but since there is one more table to spare (C3), it should be used.

Note that in the inconceivable event that all 256 byte values are equally represented in the file, each character translates into an eight-bit code. But even in this case, some compression may be obtained through the type C automatic elimination of successive duplicate byte values.

Field Type GA—Garbage Data (Permanently Unused Fields)

Field type GA is specified when the content of a field is no longer of value in the file. This type may be specified for permanently unused fields, fillers, alignment bytes, etc. They are deleted in the compressed record, but appear as binary zeros upon re-expansion. *The user should be aware that if these fields were not originally zeros, the expanded record is not an exact replica of the original.* Redundancy checking used by SAMS:Compress does not consider type GA fields.

Consider variable-length input records, where the two low-order bytes of the IBM standard Record Descriptor Word (RDW) are always binary zeros. Those two bytes can conveniently be specified as GAF2.

All four bytes of the RDW are sometimes superfluous in defining variable-length records, since the length may be implicit in the record content. The RDW can then be defined as GAF4. The File Expansion Utility fills in the length automatically. The EXPAND subroutine supplies the length if the output record address (first parameter) and the record length address (third parameter) are the same.

Field Type L—Insert Tally of Actual Length

This type is used to insert a binary length indicator at the front of each compressed record. When issued, it must be the first specification of the record definition state-

ments. In the compressed record it appears as a two-byte binary number at the start of the record after the RDW, if any, and before the type N fields. The type L field contains the number of bytes in the record which follow this field. It is coded in the record definitions simply as the letter L, with no length descriptor.

Type L is useful in reading and writing compressed records in COBOL. For example, the COBOL record definition (or redefinition) for records of maximum length 1000 can be specified as:

```
01  CMP-RCD SYNC.  
02  LEN    PIC 9(4) COMP.  
02  CHARS PIC X OCCURS 1000 TIMES DEPENDING ON LEN.
```

Field Types MA and MB—Pattern Matching

Note: The use of MA and MB pattern matching causes an 0C4 ABEND. (MA and MB pattern matching can be used with the SAMS:Compress/2 utility.) Since FDTs were converted from sequential to load modules and linked as re-entrant, the additional storage address cannot be stored within the re-entrant FDT. This problem occurs in SAMS:Compress for MVS and SAMS:Compress for IMS. For this reason, pattern matching should not be used.

When data field content is similar from record to record, a pattern matching specification may produce more efficient compression than other field type specifications.

Fields defined by this field type are compared with a pattern, and matching characters, as well as character repetitions, are compressed. Type MA fields use the data in the first record as a pattern. Type MB fields use the data in the previous record as a pattern. When SHRINK or EXPAND is called from the user's program, these patterns are set during the initial CALL. Type MB patterns are reset during each subsequent CALL.

Certain unavoidable restrictions apply to the use of these fields:

- They must be fixed-length.
- They cannot be specified within condition groups.
- When they appear within a repetition group or a nest of repetition groups, none of the enclosing repetition factors may be the variable symbol, VS.
- The pattern used to expand the field must be the same as that used to compress the field.

The use of type MB requires some further clarification. In general, the compression obtained with type MB is better than with type MA, but its usage is more restrictive. MB can only be conveniently used for sequential processing, e.g., old master updated by transaction file yields new master. The following rules must be followed for this type of sequential update:

- Each old master record must be passed to the EXPAND subroutine in sequence. This requirement precludes the use of MA or MB field definitions with ISAM files retrieved randomly.
- Each new master record must be passed to the SHRINK subroutine in sequence.
- If FDTs are in sequential data set format, the old master file and the new master file must have separate TABLxx DD statements. If the FDTs are in load module format, each master file must have its own SCB. However, these can, and usually do, refer to the same File Descriptor Table. Pattern matching works best when field values are partially or wholly repeated from record to record, as in a file containing multiple consecutive records with the same name and address. It is useful for print files, files that are transmitted, and seldom updated read-only files where the speed of compression—and especially expansion—is more important than compression ratios.

This type involves substantial set-up timing overhead per field and should, therefore, not be used for fields shorter than 10 bytes, unless a considerable compression payoff is expected. The longer the field, the less processing overhead for setup is required per byte.

Field Type N—Fields Exempted From Compression

Type N fields are not compressed. They are placed at the front of each record (after the RDW, if any, and after the type L field, if any) in the same order as they are defined. Type N fields can be used as control fields for sorting, retrieving or updating the compressed record.

Type N fields are exempted from the redundancy check byte calculation. Thus, they can be modified within the compressed record without a “Check Byte Mismatch” condition occurring upon re-expansion.

Type N fields cannot appear within conditional or repetition groups. Type N fields must appear at fixed offsets from the start of the record, and their definition must apply to all records in the file. If such a field happens to occur in the record after variable-length fields, variable repetition groups, and/or condition groups, the type N field must be defined before these other fields through the use of a Position Function, which is described later in this section. The total length of all type N fields must not exceed 4095 bytes.

The decision as to whether or not to exempt a field from compression depends upon several factors. Key fields used to retrieve records from the file must be exempted from compression to enable record retrieval. Sort key fields, upon which the file is regularly sorted, and match key fields, for record matching applications, should also be exempted from compression.

When sort key fields are exempted from compression, users can invoke a sort utility program to sort the file in its compressed state, avoiding all expansion overhead. When match key fields are exempted from compression, users can avoid expansion overhead in application programs until it is determined that compressed fields from the record require processing. Otherwise, calls to PUFFUP and PUFFDOWN can be used, saving considerable processing overhead.

Finally, any field at a fixed offset from the record origin which appears in all records of the file can be considered for exemption from compression. Consideration must be given to the trade-off between compression ratio and processing overhead. If the field is always or frequently operated upon in application programs, the user may wish to exempt the field from compression. Where compression ratio is critical, the field should be compressed. Where minimum processing overhead is critical, the field should probably be exempted from compression. Benchmark testing both ways enables the user to determine the optimal trade-off.

Field Type PD—Packed Decimal Data

Type PD fields contain packed decimal data (USAGE COMPUTATIONAL-3 for COBOL users). The field length must be less than nine bytes. Valid fields must meet the following conditions:

- The number is between -2147483647 and 2147483647.
- The sign is a hexadecimal C, D, E, or F.
- Only decimal digits (0-9) occur. (For efficiency, type PD should not be specified for fields of lengths 1 or 2. Types C1, C2, C3, UN, X, or S may be used in these cases.)

Invalid fields are automatically treated as type UN by SAMS:Compress. Instead of being compressed, a field defined as type PD which contains invalid data is enlarged by one bit. No data is lost, and upon expansion, field content is the same as it was prior to compression. If, across the file, a field is thought to contain invalid data a substantial number of times, greater compression and reduced processing overhead results from defining the field as a type C instead of PD. A message is printed with the statistics produced by the File Compression Utility indicating the number of times invalid data was encountered in a PD field during compression.

Packed decimal numbers are converted by SAMS:Compress to binary, bit aligned, variable-length “floating point”. If the packed decimal numbers are large in magnitude and fill their fields with significant digits (e.g., the packed date 76 36 5C in a three-byte field), then defining the field as type PD performs poorly in both average time per byte

and compression. It is better to specify a type C dedicated to these packed decimal fields. However, if the numbers rarely come close to filling the field with significant digits, specifying type PD yields improved performance over alternative specifications.

Specifically, fields with value 0 compress to 6 bits, while fields with value between $16n-1$ and $16n-1$ compress to $6+4n$ bits, regardless of sign or field width.

Consider a file where each record consists of 20 four-byte packed fields, either written as PDF4, PDF4, ..., 20 times, or abbreviated as a repetition group

```
20'PDF4'.
```

If the average number of significant digits is five or greater, then it is better to define the record as

```
C1F80.
```

If compression is more important than speed, then specifying

```
20'C1F3,C2F1'
```

separately defining the low-order sign bytes gives better results for the same record.

Field Types S and X—Set of Expected Values

A table reference compression technique can be used where the set of expected values contained in a field across the file is small, and these values are known. It is not necessary to include all values that occur in the field in the table of expected values. If data is encountered in the field for which no matching table entry is specified, the data is not compressed; instead, it grows by one bit. No data is lost in this case, and the expanded field is identical to the field before compression. A message is printed with the statistics produced by the File Compression Utility, indicating the number of times a value was encountered in a type S or X field that was not specified in the table of expected values. To achieve efficient compression, it is important that most values occurring in a field defined as type S or X are specified in the table of expected values.

Field types S and X are functionally equivalent. The only difference is in the method of specifying the table of expected values. The table for type S is coded in EBCDIC characters. Any of the 256 possible byte values can be coded, but non-graphic data must be multipunched. The table for type X is coded in hexadecimal format. Each byte value in the table is coded as two hexadecimal digits.

RDL specifications for field types S and X are coded in a special format:

```
tmmv
```

where:

- t* is the field type specification, either S or X.
- m* is a two-digit number ($01 < m < 99$ — code the leading zero for values between 01 and 09), indicating the field length.

- n* is a two-digit number (01<*n*<16 — code the leading zero for values between 01 and 09), indicating the number of entries in the table of expected values.
- v* is the table of expected values. Table entries of expected values are coded consecutively until *n* values are specified. For type S, no space can be left between consecutive entries. The table must occupy exactly *m***n* positions in the field definition. If the table specification continues past column 72 of the current RDL statement, it must be continued starting in column 1 of the next RDL statement. For type X, spaces may appear between pairs of hexadecimal digits for readability, but the table must contain exactly 2**m***n* hexadecimal digits.

SAMS:Compress uses a sequential search algorithm to determine if a field value in the record appears in the table of expected values. For maximum efficiency in processing overhead, the entries of the table should be coded in decreasing order of probability of occurrence. The expected value most likely to occur should be coded first; the expected value least likely to occur should be coded last.

The only limit on the size of an expected value table, other than 99 entries maximum, is the total space available in the FDT. Where applicable, this is the most efficient method, both in terms of compression ratio and processing overhead.

To illustrate correctly coded type S and X field definitions, and to illustrate the difference in the way expected values are coded between types S and X, consider the following definitions, which are equivalent:

```
S0103AB1
X0103C1C2F1
```

Suppose a file has a four-byte field containing DOGb/, CATb/, FISH, BIRD, FROG, or “other”, where “other” occurs infrequently. If this field is specified as:

```
S0405DOGbCATbFISHBIRDFROG
```

the 32-bit (i.e., four-byte) field compresses to four bits, one bit as an error flag and three bits to represent which of the five values occur. An “other” field cannot be compressed, but grows by one bit to 33 bits.

Field Type UN—Undefined Fields

This type is used for fields which fall into none of the other categories. Its chief use is to define fields which cannot readily be compressed (floating point, binary, bit switches, etc.), particularly when types C1, C2 and C3 are already in use.

For example, consider defining a floating point field, and types C1, C2 and C3 are already in use. To define the floating point field as one of the C types alters the distribution of values in the corresponding character frequency table. Defining a floating point field as a type C has a detrimental effect upon the compression ratio for all of the other fields defined by the type C specification. To avoid this effect, define the floating point field as type UN. Type UN fields are not compressed, but neither do they grow in length. Processing overhead for type UN fields is minimal.

Field Types V, VP, and VZ—Calculating a Value for the Variable Symbol

RDL provides the capability for defining variable-length fields, and fields which occur a variable number of times, if the length of the variable-length field, or the number of times a variably occurring field is actually present, is stored within the record or can be calculated from information stored within the record. This capability is implemented using the Variable Symbol, which is coded in RDL specifications as VS.

Field types V, VP and VZ provide the means to calculate and store a value in the VS. VS is then coded in subsequent RDL specifications as a field length, a position reference or a repetition factor. The value stored in the VS during file processing is substituted in the RDL specification in which it appears for each record for which the specification applies. The VS may be referenced multiple times within the definition of the record. The value stored in the VS is changed every time a type V, VP or VZ field is processed.

Field types V, VP and VZ are functionally equivalent. The only difference is in the format of the data contained in the type V, VP or VZ field. Use type V to define fields which contain binary integer data, type VP to define fields which contain packed decimal data, and type VZ for fields which contain right-justified zoned decimal data.

RDL specifications for field types V, VP and VZ are coded in one of three possible special formats at the user's discretion:

```
t n
t n o1 i1
t n o1 i1 o2 i2
```

where:

t is the field type specification, either V, VP, or VZ.

n is the length of the V, VP, or VZ field, in bytes.

For type V: 1 < n < 4.

For types VP and VZ: 1 < n < 8.

o₁ and o₂ + = addition

- = subtraction

* = multiplication

/ = division

i₁ and i₂ are integers between 0 and 32767.

Note that any remainder resulting from a division operation is dropped.

Arithmetic operations are evaluated left to right. The following are examples of the special formats:

V4 means that the four-byte field currently defined contains a binary integer whose value is to be stored in the VS. Use of the VS in a

subsequent RDL specification refers to the value of the binary integer.

V4+500 means that 500 is added to the binary integer, as described in the previous example, and the result is stored in the VS.

VP3-3/20 means that the three-byte field currently defined contains a packed decimal number. Three is subtracted from the number, and the result is divided by 20, and the end result is stored in the VS.

Processing variable-length records is frequently encountered as a use for type V specification. For example, for a variable-length record that is treated as a single type C1 field, the record definition is written as

V2-4,GAF2,C1FVS.

The record length is picked up from the first two bytes of the RDW, from which the RDW length, 4, is subtracted. The next two unused bytes of the RDW are treated as a garbage field, while the remainder of the record is character data.

Consider a variable-length record made up of the four-byte RDW, followed by a fixed 80-byte field that is followed by a variable number of 40-byte appendages. The fixed portion is treated as type C1, while each appendage contains 10 four-byte packed decimal numbers. The record is defined as

V2-84/40,GAF2,C1F80,VS'10'PDF4''.

One restriction applies to type VP and VZ fields. If invalid packed decimal data is encountered in a field defined using VP, or invalid right-justified zoned decimal data is encountered in a field defined using VZ, SAMS:Compress ABENDs with a user code of 20, and the following message is written to the system output writer:

```
VP
INVALID TYPE VZ FIELD
```

Coding the RC parameter when calling the SHRINK or EXPAND subroutine suppresses the ABEND, as shown with the messages:

```
REC DEFS IMPLY WRONG LENGTH
```

and

```
CHECK BYTE MISMATCH
```

(See the “SAMS:Compress/2” chapter and the “Messages and Abend Codes” chapter for further information.)

Field Types ZL and ZR—Zoned Decimal Data

Field types ZL and ZR can be used to define fields which contain zoned decimal data. Type ZL defines a field containing left-justified zoned numeric data, possibly followed by one or more filler characters. Type ZR defines a field containing right-justified zoned numeric data, possibly preceded by one or more filler characters. Valid fields must meet these conditions:

- The numeric portion must contain only digits (0-9). Commas and decimal points are not permitted.
- The value in the numeric portion must be less than 2147483648. Negative numbers are not permitted.
- The length of the field must not exceed 128 bytes.
- For ZL fields, the filler character, if any, must be blank. All-blank fields are valid, as is a field containing a single, left-justified zero, followed by all blanks.
- For ZR fields, the filler character may be either blank or zero, but not both. All-blank and all-zero fields are valid. A field containing a single, right-justified zero preceded by all blanks is valid.

Data contained in fields defined as ZL or ZR are converted to binary, bit aligned, variable-length “floating point”. Blank or zero fields compress to five bits, while fields with value between 16^{n-1} and $16^n - 1$ compress to $5 + 4n$ bits, regardless of field length.

If invalid data is encountered in a field defined as ZL or ZR, the field is not compressed. Instead, it grows by one bit in the compressed record. No data is lost in this event; the expanded field contains exactly the same data as it did before compression. A message is printed with the statistics produced by the File Compression Utility indicating the number of times invalid data was encountered in a ZL or ZR field during compression.

ZL and ZR specifications are most useful in cases where a separate type C specification cannot be dedicated to zoned numeric fields, and/or multiple zoned numeric fields are not contiguous in the record. Consider an 80-byte record containing all numeric digits. The most efficient specification is C1F80. However, if 80 contiguous bytes containing zoned numerics occurred within a record for which all type C definitions were already in use, the best definition for the numeric data is 8'ZRF9',ZRF8. For zoned decimal fields of nine bytes or less, a definition using type ZR or ZL is preferable to using a type C specification.

RDL Repetition Groups

If a sequence of one or more field definitions is repeated n times, it may be coded once with a “repetition factor” by enclosing the sequence within single quotes and preceding the initial single quote with a two-digit number, n , where $02 < n < 99$. For example:

```
02'ZRF2,03'PDF4,C1F25''.
```

is an abbreviated form of

```
ZRF2,PDF4,C1F25,PDF4,C1F25,PDF4,C1F25,
ZRF2,PDF4,C1F25,PDF4,C1F25,PDF4,C1F25.
```

Repetition groups may be completely, but not partially, contained in conditional groups and vice versa. Thus:

`... (...03'...)...`

and

`03'... (...'...)`

are syntactical errors.

The symbol VS can be used instead of the two-digit repetition factor to indicate that the value used is specified in a previous type V definition. VS is useful when the actual number of times that a field or a series of fields occurs within the record is variable and is contained in a separate field. For example, an invoice file consisting of variable-length records has a variably occurring series of three fields:

1. Line item description, 20 bytes of character data
2. Line item quantity, four bytes packed decimal data
3. Line item amount, seven bytes packed decimal data

A separate field in the record contains a value indicating the number of line items represented in the record. This field is two bytes in length and is zoned decimal format. The following is coded to define these fields:

`...VZ2,...,VS'CLF20,PDF4,PDF7' etc.`

where

VZ2 is the field containing the actual number of occurrences

VS is the Variable Symbol reference used to refer to the value (contained in the previously defined V-type field) which contains the actual number of occurrences

and the characters enclosed in single quotes represent the line item fields.

RDL Condition Groups

SAMS:Compress's RDL provides the capability for defining multiple record formats for a file, where the format of an individual record can be determined from the contents of a field within the record by coding RDL specifications in a special structure, the condition group.

The general form of a condition group follows:

`(nv, f, ..., f)`

where:

n is a two-digit number, indicating the length of the field that is tested.

v is a value for comparison, *n* bytes in length.

f is any RDL specification or repetition group.

If the current *n* bytes in the input record being processed are equal to the value *v*, then the remaining definitions within the parentheses apply to the record; otherwise, they are skipped.

The value coded for *v* can consist of any of the 256 possible byte values; however, non-graphic values must be multipunched.

To avoid excessive multipunching, users can code the value *v* in hexadecimal format. To do so, code an X preceding the length specification *n*, and code the value as pairs of hexadecimal digits. For readability, spaces may be left between pairs of hexadecimal digits. For example, the following two condition group specifications are equivalent:

(03XYZ,C1F80)

(X03E7E8E9,C1F80)

A series of consecutively coded condition groups which are not separated by a comma (i.e., separated by one or more blanks) indicates that the first condition group in the series whose condition is met applies to the record being processed, and the remaining condition groups in the series are skipped. For example, consider a hypothetical elementary invoicing file. Assume this file consists of sets of records, where each set of records represents one invoice. Each set of records consists of a header record, one or more detail records and a trailer record. Figure 7-1 illustrates these records and the fields contained in them.

INVOICE #	RECORD TYPE	INVOICE DATE	filler		RECORD TYPE=H
"	"	ITEM DESCRIPTION	ITEM QTY	ITEM AMOUNT	RECORD TYPE=D
"	"	"	"	"	RECORD TYPE=D
"	"	filler	TOTAL QTY	TOTAL AMOUNT	RECORD TYPE=T
Field	Length	Data Type			
INVOICE #	8	Zoned Decimal			
RECORD TYPE	1	Character (H, D, or T)			
INVOICE DATE	8	Alphanumeric			
ITEM DESCRIPTION	20	Characters			
ITEM QUANTITY	4	Packed Decimal			
ITEM AMOUNT	7	Packed Decimal			
TOTAL QUANTITY	4	Packed Decimal			
TOTAL AMOUNT	7	Packed Decimal			

Figure 7-1. Invoicing File Record Set and Field Descriptions

The following RDL specifications define these records and illustrate the use of condition groups:

```
ZRF8, (01H,C1F8,GAF23) b / (01D,C2F20,PDF4,PDF7) b (01T,GAF20,PDF4,PDF7) .
```

It is important that alternative condition groups in a series be separated from one another by one or more blanks, and not a comma. The comma is used to separate the last condition group in a series from any RDL specifications (which could be another condition group) which follow.

To illustrate this point, consider the sample RDL specifications above, describing the records in Figure 7-1. Suppose a comma separated the first condition group from the second condition group. Then if a record whose RECORD TYPE field contained an H was encountered, the first condition group applies. But after processing the C1F8,GAF23 RDL specifications, the following condition groups are not skipped. SAMS:Compress expects more data beyond the “filler” at the end of the record, looking to compare for a “D”, according to the next condition group. Since the record definitions do not accurately define the record that is processed, unpredictable results can occur.

Note that if the file contains a record whose record type is not H, D or T, SAMS:Compress will ABEND with a user code of 15 and the message “REC DEFS IMPLY WRONG LENGTH.” This is because no record definitions are specified past the invoice number for any record whose record type is neither H, D nor T. The user can avoid this situation by coding a default condition group at the end of the condition group series.

It is permissible (and frequently necessary) to code a default condition group at the end of a series of condition groups. Such coding supplies a set of record definitions in the event that none of the preceding condition group tests are met, and the actual content of the byte(s) in the record which is being tested is not known. The general form of a default condition group is the following:

```
(00,f, . . . ,f)
```

where:

00 is coded exactly as shown.

f is any RDL specification or repetition group.

A condition group coded in this form means that no matter what the contents of the current byte are in the record being processed, the RDL specifications in the condition group apply. Any default condition group coded in a series of condition groups must be coded as the last in the series. To illustrate this procedure, the following definition can be coded to describe the records in Figure 7-1:

```
ZRF8, (01H,C1F8,GAF23) b (01D,C2F20,PDF4,PDF7) b (01T,GAF20,PDF4,PDF7) b  
(00,C3F32) .
```

If the user omits coding a default condition group as the last condition group in a series, SAMS:Compress automatically supplies the default condition group “(00)”. This procedure indicates that if none of the preceding condition groups apply for a particular

record, any RDL specifications following the condition group series apply beginning at the current byte location in the record. This is the same byte location within the record which the preceding series of condition groups tested. To illustrate this point, the user should note that the following two sets of RDL specifications are equivalent:

```
ZRF8,(01H,C1F8,GA23)⊕(01D,C2F20,PDF4,PDF7)⊕(01T,GA20,PDF4,PDF7)⊕
      (00,C3F32).
ZRF8,(01H,C1F8,GA23)⊕(01D,C2F20,PDF4,PDF7,)⊕(01T,GA20,PDF4,PDF7),
      C3F32.
```

The maximum number of condition groups which may be coded in a series is 16, including the final default condition group, whether user-specified or automatically provided by SAMS:Compress. The maximum number of condition group series which can be coded is limited only by the amount of space available in the FDT.

If the condition specified in a condition group is met, the value specified in the condition group and found in the record is compressed to four bits, regardless of the length of the value. No separate RDL specification is used to compress the value. The lengths of the values may differ within a condition group series.

RDL Position Function

As fields are processed, SAMS:Compress automatically adjusts an internal field pointer (IFP) to the current displacement within the record. In a few special cases, the user may need to alter this IFP with the Position Function.

For example, fields that are exempted from compression (i.e., fields defined with the field type-N RDL specification) must be defined before any variable or condition group RDL specifications. If a field that is exempted from compression is actually located at a higher displacement from the record origin than variable-length fields or conditionally present fields, the Position Function must be used to set the IFP at the field that is exempted from compression, so that it can be defined first. Then the Position Function must be used again to reset the IFP to the lower displacement so that the variable-length and/or conditionally present fields can be defined.

The Position Function has four possible forms, chosen at the user's discretion:

Pn	Set IFP to n .
$P+n$	Add n to the IFP.
$P-n$	Subtract n from the IFP.
P	Reset IFP to prior value.

P is coded as shown; n is either a one- to five-digit integer or the Variable Symbol, VS.

Displacements are computed relative to 0, which indicates the start (origin) of the record. The four Position Function forms perform the following functions:

P_n repositions to the $n+1$ th byte in the record.

$P+n$ repositions forward n bytes.

$P-n$ repositions backwards n bytes.

P resets the IFP to its value immediately preceding the last P_n , $P+n$ or $P-n$. If there were no previous Position Functions executed, the P is ignored. P cannot be specified as the initial Position Function. It is the user's responsibility to adhere to the following rules:

- The IFP must remain within the bounds of the record. This error is usually, but not always, caught during the prepass or compression phase, because complete checking would involve substantial timing overhead during compression.
- The IFP must be at the byte following the last byte of the record after all fields are processed. Otherwise, a wrong length record ABEND 15 occurs.
- If a field is bypassed, a check byte mismatch ABEND 10 can occur on re-expansion.
- Redefining through repositioning degrades performance and should be specified only when absolutely necessary.

For example, a hypothetical name and address file contains three types of 80-byte records as shown in Figure 7-2.

NAME	RECORD TYPE	RECORD TYPE=A
STREET ADDRESS		RECORD TYPE=B
CITY, STATE	ZIP CODE	RECORD TYPE=C
<u>Field</u>	<u>Length</u>	<u>Type of Data</u>
NAME	79	Character
RECORD TYPE	1	Character
STREET ADDRESS	79	Character
CITY, STATE	74	Character
ZIP CODE	5	Zoned Decimal

Figure 7-2. Name and Address File Record Set and Field Descriptions

The following RDL specifications define this file:

`P79,(01A,P,C1F79)⋮(01B,P,C2F79)⋮(01C,P,C3F74,ZRF5),P+1.`

P79,	sets the IFP at the RECORD TYPE field. (Note that the IFP is relative to zero, thus IFP of zero is the first record position, and IFP of 79 is the 80th byte of the record.)
(01A,P,C1F79)⋮	If the RECORD TYPE field contains A, resets the IFP to the beginning of the record, and defines the NAME field.
(01B,P,C2F79)⋮	If the RECORD TYPE field contains B, resets the IFP to the beginning of the record, and defines the STREET ADDRESS field.
(01C,P,C3F74, ZRF5),	If the RECORD TYPE field contains C, resets the IFP to the beginning of the record, and defines the CITY, STATE and ZIP CODE fields.
P+1	At this point, one of the condition groups has been applied to the record (provided, of course, that all records in the file contain either A, B or C in the RECORD TYPE field), and thus the IFP is pointing, once again, at the RECORD TYPE field. Coding this specification, P+1, is necessary to position the IFP at the byte location following the last byte of the record. Failure to do so results in an ABEND with a user code of 4 and the message, "REC DEFS IMPLY WRONG LENGTH."

General Restrictions on RDL Use

The Record Definition Language is employed to construct the File Descriptor Table (FDT). The FDT has a maximum size and, as such, there is a corresponding upper limit on the maximum amount of RDL specifications permitted for any one file. In the unlikely event that this maximum is reached, a user ABEND 4 occurs and the RDL specifications must be reduced by combining adjacent field definitions to form group fields.

The space in the FDT required to contain the RDL specifications can be calculated using the table in Table 7-3.

Table 7-3. FDT Space Requirements of RDL Specifications

RDL Specification	Space Required (In Bytes)
GA, PD, ZL, ZR	6
C1, C2, C3	18
MA, MB	92 for first occurrence. 54 for each subsequent occurrence
UN	14
N	20
V	48
VP, VZ	40
S, X	26
First "(" in a condition group series	72
Each remaining "(" in a condition group series	54
Fn, where n < 128	4
Fn, where n > 128	38
FVS	46
VER, Dc	88
P	12
P + n	18
P + VS	8
Fixed repetition group	16
Variable repetition group	30

The space required must be summed according to RDL specifications as coded, and the total may not exceed 2800. For example, consider the following RDL specifications:

V2-4, GAF2, C1FVS.

Using the table, the space required for these RDL specifications is:

V2-4	48
GA	6
F2	4
C1	18
FVS	46
	122 (total)

This is well within the 2800 limit.

Guide to Correct RDL Specifications

The following is a list of field types with the corresponding description beneath each type. The numbers in parentheses that follow each description are a relative measure of processing efficiency with “1” being the most efficient, and “6” being the least efficient.

C1, C2, C3

Type C is used to define groups of fields whose byte values have similar frequency distributions. Up to three different frequency distributions can be accommodated, one each by type C1, C2 and C3. If no other type code is clearly preferable, choose a type C.

Compression is variable, depending on the skew of the distribution. The greater the skew of the distribution, the greater the compression ratio. Processing overhead is minimal (3).

GA

Type GA is used to eliminate unneeded fields from the compression record. Compression is 100 percent. Processing overhead is negligible (1).

L

Type L is used to insert a binary tally of the compressed actual number of bytes comprising the compressed record as the first two bytes of the record following the RDW. Particularly useful for COBOL users. This field type actually increases the compressed record length by two bytes. Processing overhead is negligible (1).

MA, MB

Type M is used when data in a field repeats from record to record. Several restrictions govern use of this field type. For fields smaller than 10 bytes in length, type C is preferable.

Compression is variable, depending upon the degree of data repetition. Processing overhead is variable, decreasing as field length increases (3-5).

N

Type N is used to exempt a field from compression. Use for retrieval, match and sort keys, and any frequently processed field where exempting the field from compression frequently enables use of PUFFUP and PUFFDOWN instead of SHRINK or EXPAND subroutines. There are several restrictions governing use of this field type specification.

This field type yields no compression. Processing overhead is negligible (1).

PD

Use type PD for fields containing packed decimal data, preferably with many high-order zero digits. If significant digits frequently fill the field or if invalid data is frequently present in the field, choosing a type C specification is preferable.

Compression is excellent when the value is zero, and variable, increasing as the proportion of significant digits to total digits decreases. Processing overhead is moderate (5-6).

S, X

Use type s or x when the number of values occurring in a field is small, sixteen or fewer, and these values are known in advance.

Compression is excellent, and the processing overhead is minimal (3).

UN

Use Type UN for fields which cannot readily be compressed (e.g., bit switches, floating point numbers), particularly when all three type C specifications are already in use.

This field type yields no compression. Processing overhead is minimal (2).

V, VP, VZ

Use one of these field type specifications to define a field whose content is used to calculate the actual length of a variable-length portion of the record, or a field which contains the actual length of a variable-length portion of the record.

Type V is not compressed. Type VP is compressed as PD. Type VZ compressed as ZR. Processing overhead is minimal to moderate, depending on field type:

(V=2, VZ=4, VP=6)

ZL, ZR

Type Z is used for fields containing zoned decimal data, particularly when a type C specification cannot be dedicated to zoned decimal data. Several restrictions govern the use of this field type specification.

Compression is excellent when value is zero, variable, increasing as the magnitude of the value increases. Processing overhead is moderate (3-4).

RDL Defaults

If the RECDEF DD statement is not present in the execution JCL for the File Prepass Utility or by the Interactive Dialog, or if it specifies a null data set (i.e., one with no records), default RDL specifications are generated based upon characteristics of the data set defined by the INFILE DD statement. The defaults are also generated if the only RDL specification supplied by the user is a single type L field.

In the default RDL specification formulas shown in Tables 7-4, 7-5 and 7-6, the following variables are substituted with appropriate values, obtained from the data set label or JCL specifications:

x	number of bytes prior to the key, excluding the RDW (if present)
x'	x-1
y	number of bytes following the key; if there is no key, then y is the record length (LRECL)
k	number of bytes in the key (KEYLEN)
k'	k+1
j	k + the relative key position (RKP)

The generated defaults are printed on the PRINT data set by the File Prepass Utility or the Interactive Dialog.

Table 7-4. RDL Default Specifications for Sequential Files

Files	Formulas for Default RDL Specifications
Fixed-length	C1Fy
Variable-length	V2-4,GAF2,C1FVS

Table 7-5. RDL Default Specifications for ISAM Files

File	Formulas for Default RDL Specifications
Fixed-length, key at beginning of record	Nk,C1Fy
Fixed-length, RKP=1*	Nk',C2Fy
Fixed-length, RKP>1	*N1,C1Fx',Nk,C2Fy
Fixed-length, key at end of record*	N1,C1Fx',Nk
Variable-length, relative key position = 4	V2-j,GAF2,Nk,C2FVS
Variable-length, RKP=5*	V2-j,GAF2,Nk',C2FVS

Table 7-6. RDL Default Specifications for VSAM Files

File	Formulas for Default RDL Specifications
Fixed-length, key at beginning of record	Nk,C1Fy
Fixed-length, key somewhere within the record	N1,C1Fx',Nk,C2Fy
Fixed-length, key at end of record	N1,C1Fx',Nk
Variable-length, key at beginning of record	Nk,C2VER
Variable-length, RKP=1*	Nk',C2VER
Variable-length, RKP>1*	N1,C1Fx',Nk,C2VER

Notes for Tables 7-4, 7-5, and 7-6:

* Default definition permits record deletion when the DCB parameter OPTCD=L is specified.

If "L." is specified, all defaults begin with "L,...".

Chapter 8. Implementing SAMS:Compress Without the Interactive Dialog

This chapter provides instructions to SAMS:Compress for IMS users who are not able to use the facilities of the Interactive Dialog (a supplied ISPF interface) to implement SAMS:Compress for IMS. For these users, SAMS:Compress for IMS is implemented using SAMS:Compress utilities.

The “Installation” chapter of this manual provides step-by-step instructions for installing the SAMS:Compress family of products. If you are not using the Interactive Dialog, skip those sections dealing with its installation. However, you must follow all of the instructions for installing the SAMS:Compress for IMS interface, and the SAMS:Compress/2 product.

The main SAMS:Compress for IMS chapter of this manual provides a detailed product description and contains other reference materials which are applicable regardless of the method used to implement the product. Users should become familiar with the information presented there before trying to implement SAMS:Compress for IMS manually through the SAMS:Compress utilities as described here.

Once SAMS:Compress for IMS has been implemented, it is completely transparent to IMS applications programs. The applications programs require only minor JCL additions (i.e., STEPLIB DD statements to FDT library and SAMS:Compress for IMS modules) to access compressed data base segments.

Implementation Overview

SAMS:Compress for IMS is implemented in two phases:

1. Data Base Prepass—The data base segments to be compressed are defined to SAMS:Compress with the standard Record Description Language. The SAMS:Compress for IMS utility, IMSPASS, is executed (as an IMS application program) to generate a File Descriptor Table for each segment type to be compressed.
2. Data Base Compression—The data base to be compressed is unloaded, the SAMS:Compress for IMS Interface, IMSHRINK, is specified as the compression routine in a DBD generation and the data base is reloaded. Data base reload invokes the user exit, IMSHRINK, resulting in a data base with compressed segments.

IMS/VS execution subsequently invokes IMSHRINK (based upon DBD specification) for retrieve/insert/replace/load operations, as shown in Figure 8-1.

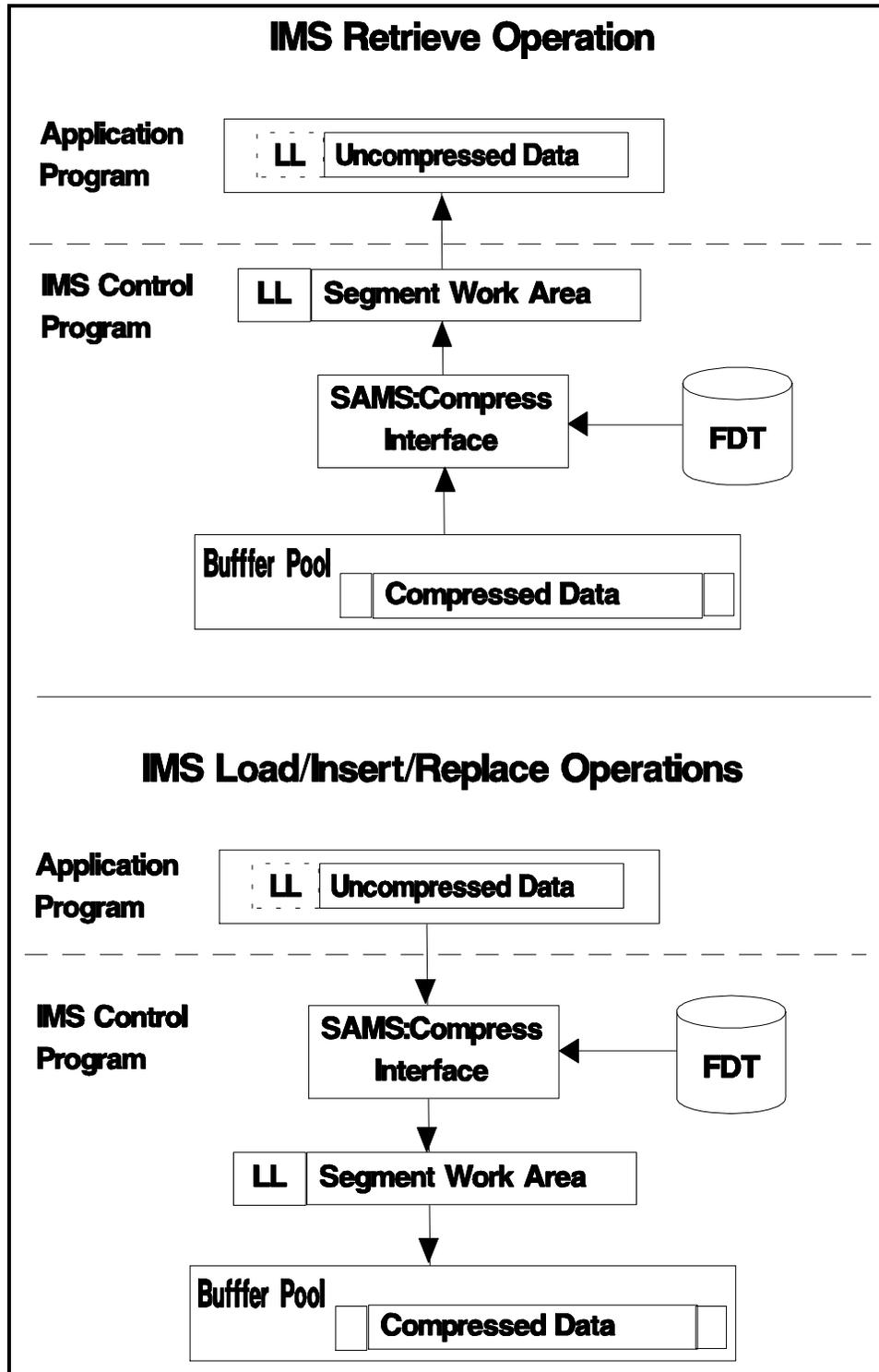


Figure 8-1. SAMS:Compress for IMS Operation

Obtaining Optimal Compression

SAMS:Compress provides a powerful Record Definition Language (RDL) for the user to describe segment (record) characteristics and to specify content-oriented compression techniques desired. Each segment type can be defined in fine detail (to the field level or less). The level of detail chosen depends on the knowledge of segment content and an evaluation of the trade-off between compression and overhead.

Each segment type is first described to the SAMS:Compress for IMS prepass program, IMSPASS. Simplified use is made of the RDL, as described in “Describing Segments with RDL” below. Trial compression statistics are then analyzed, and a determination is made regarding whether or not coding RDL compression specifications in finer detail is required to obtain better compression.

If further prepass/compression is desired, IMSPASS can be re-executed with different RDL. Alternately, a sequential data set containing a user-specified sample of segments (optional output from IMSPASS) can be input to the standard SAMS:Compress system. This option provides execution efficiency over IMS/VS access of the entire data base with IMSPASS.

User Constraints

General constraints that apply to use of SAMS:Compress for IMS are:

- The PSB referenced in IMSPASS cannot show PL/I as the program language.
- Compressed segments may not be processed by a user-supplied edit exit; i.e., only one COMPRTN allowed per SEGM macro—an IMS constraint.
- The entire segment sequence field (from the start of the segment through the end of the key) must be left uncompressed. This is done through the type N RDL specification.

Data Base Integrity

SAMS:Compress for IMS is designed to minimize the loss of data base integrity or the possibility of abnormal termination in a production environment:

- SAMS:Compress for IMS includes self-checking and error analysis features to trap specification errors during execution of the IMSPASS utility, DBD generation or the IMS reorganization/reload utility.
- SAMS:Compress for IMS includes redundancy checking logic to verify that each expanded record (segment) is identical to the record before compression.

Assuming initial data base integrity and error-free application programs, further errors should not occur.

Defining the Data Base to SAMS:Compress for IMS

Procedural Overview

Each segment type to be compressed must be defined to SAMS:Compress for IMS by:

- Specifying the characteristics of the segment type using the Record Definition Language. The segment key fields are described as fields which are not compressed. The remainder of the segment type is described to whatever extent the user desires in order to balance compression obtained and execution performance.
- Executing the IMSPASS utility. This utility creates the File Descriptor Table (FDT) for the segment type. The FDT contains compression/expansion data for use by SAMS:Compress for IMS to encode/decode a segment. If a segment shares an FDT, do not execute IMSPASS for the segment.
- Optionally, the RDL can be revised to attempt to improve compression. To permit efficient experimentation, the prepassed segments read by IMSPASS can be written to a sequential data sets. This data set can then be read by the File Prepass Utility to evaluate the effectiveness of the revised segment description and to create the FDT for the segment.

When RDL revision, if any, is completed, the FDT must be converted to load module format on the FDT library by the FDTLOADR utility.

This process is summarized in Figure 8-2. Subsequent sections describe the use of RDL, execution of IMSPASS, and execution of FDTLOADR.

Describing Segments With RDL

The Record Definition Language (RDL) is used to describe the segment types of the user data base to be compressed. RDL allows the user to describe the various data field types within a segment to indicate explicitly the compression technique to be used.

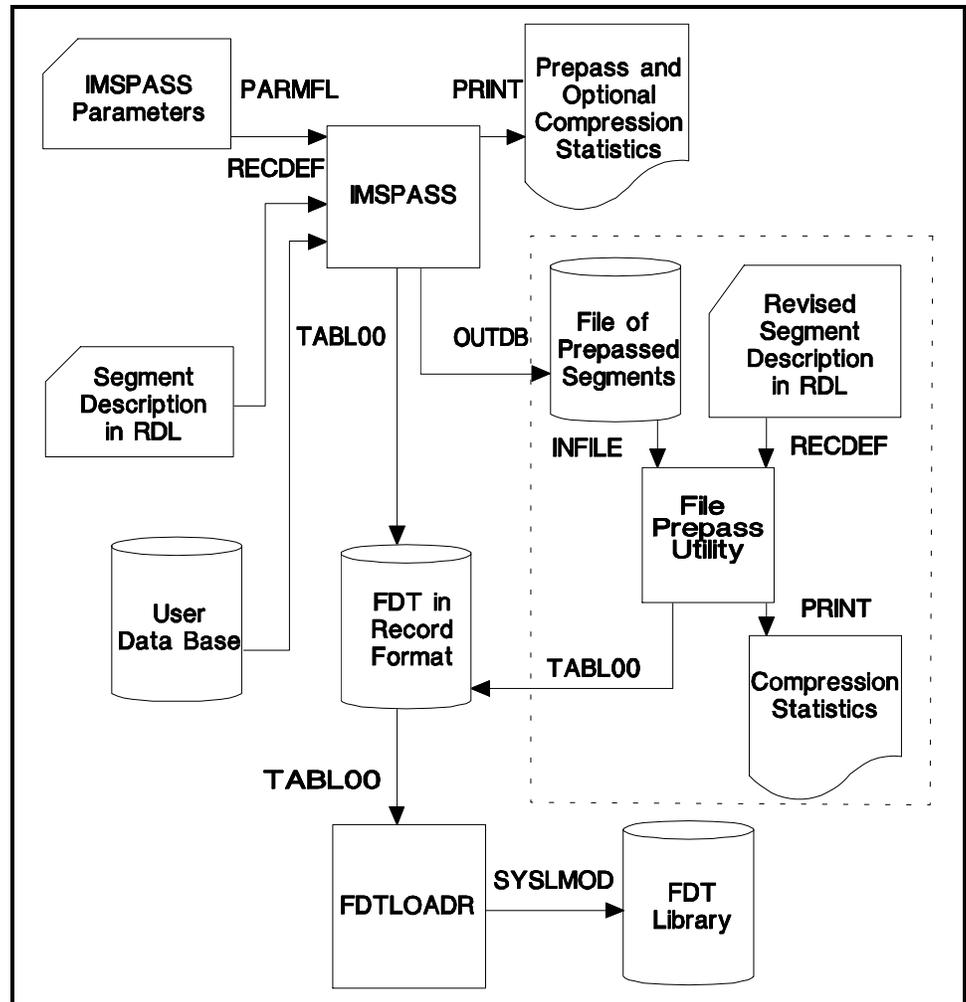


Figure 8-2. Defining the Data Base to SAMS:Compress

IMSPASS is executed once for each segment type to be compressed.

As indicated in dotted lines, the optional sequential file OUTDB may be input to the File Prepass Utility for further experimentation with the RDL to optimize compression of the segment type.

When the compression is satisfactory, the FDT is loaded onto the FDT library with the FDTLOADR Utility.

Simplified use of the RDL is illustrated in this section which should enable the user to obtain satisfactory compression. Only two basic characteristics of the segment need be defined with RDL:

1. The location and length of the segment key (sequence field) which is not to be compressed.
2. The length of the data portion of the segment which is to be compressed using the Huffman encoding technique.

The segment key is defined as a type N field (no compression); the remainder of the segment is defined as a type C field (compress using the Huffman encoding technique).

A more detailed description of the segment type may be made, if desired, using the RDL. Refer to the RDL chapter in this manual for a complete description of the Record Definition Language. All field type compression specifications are permitted by SAMS:Compress for IMS, except pattern matching (codes MA, MB).

The RDL specification for the segment type is input to the IMSPASS utility as the data set with DDNAME RECDEF. Suggested RDL specifications for fixed-length and variable-length segments follow.

Defining Fixed-Length Segments

The RDL specification for fixed-length segments may initially be coded as shown in Figure 8-3.

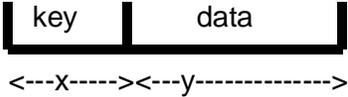
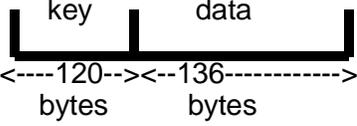
Segment Format	RDL Specification
<p>Generalized Format</p>  <p><---x-----><---y-----></p>	<p>Nx,C1Fy. Where:</p> <p>x = key length, up to 4 digits</p> <p>y = data length, specified as a value from 1 to 16384.</p> <p>If the segment is not keyed, the "N" value should be omitted.</p>
<p>Example:</p>  <p><---120--><---136-----></p> <p>bytes bytes</p>	<p>N120,C1F136.</p> <p>Specifies no compression for 120 bytes, character encoding for fixed length of 136 bytes.</p>

Figure 8-3. RDL for Fixed-Length Segments

Defining Variable-Length Segments

The RDL specification for variable-length segments includes definition of the 2-byte length indicator, using the value in the length indicator to calculate the length of the data portion. Figure 8-4 illustrates RDL coding format:

Segment Format	RDL Specification
<p>Generalized Format</p> <p>LL is a 2-byte field containing $2+x=y$, the segment length</p>	<p>V2-f,Nx,C1FVS. Where:</p> <p>f=offset of first position beyond the key (i.e., x+2)</p> <p>x = key length, up to 4 digits</p> <p>If the segment is not keyed, f=2 and the "N" value should be omitted.</p>
<p>Example:</p> <p>LL is a 2-byte field containing the segment length</p>	<p>V2-102, N100,C1FVS. Specifies 2-byte length field, no compression for 100 bytes, character encoding for variable length (calculated as segment length minus 102).</p>

Figure 8-4. RDL for Variable-Length Segments

Executing the IMSPASS Utility

A separate IMSPASS step must be executed for each segment type to be compressed. The IMSPASS utility is controlled by an input data set of parameters specifying execution options such as the number of records (segments) to process, whether or not trial compression is to be performed, and whether or not a sequential data set containing a sample of segments is to be output. It executes as a batch IMS application program. (See the "Installation" chapter for IMSPASS installation procedures.) Functions of IMSPASS are:

1. Read and edit the RDL statements describing the segment type.
2. Validate the RDL specifications by a prepass of a user-specified sample of segments.
3. Perform trial compression (optional) and calculate compression statistics for a user-specified sample of segments. Compression is performed, but the data base is not altered.
4. Generate the File Descriptor Table (FDT) for the segment type.

IMSPASS execution is controlled by an input file of keyword parameters, most likely an input stream data set. IMSPASS data sets are described in Table 8-1.

Table 8-1. IMSPASS Data Sets

DDNAME	I/O	Data Set Description
imsnames	I/O	Standard IMS JCL requirements for batch program execution, defining the data base to DFSLI000.
PARMFL	I	IMSPASS Parameter File specifying execution options. See Table 8-2.
RECDEF	I	Record Definition Language statements describing the segment type.
TABL00	O	The File Descriptor Table for the segment type prepassed. A sequential data set.
OUTDB	O	<p>Sequential output data set (optional) of prepassed segments. This file may be input to the File Prepass Utility with different RDL specifications to determine the optimal compression method.</p> <p>IMSPASS forces the following DCB specifications for this data set:</p> <p><i>RECFM=U</i> is used for both fixed-length and variable-length segments. The RDL input to the File Prepass Utility specifies the actual record (segment type) format.</p> <p><i>BLKSIZE</i> reflects the largest segment length of the data set. The IMSPASS output indicates the actual maximum length.</p>
OUTFILE	O	<p>Optional dummy data set required only if trial compression statistics are to be taken, i.e., the "C=..." parameter is specified. The DD statement is</p> <pre>//OUTFILE DD DUMMY,DCB=(LRECL=x,BLKSIZE=y)</pre> <p>Where x=maximum segment size + 8, y=x + 4</p>
PRINT	O	<p>Print data set (normally SYSOUT) to record RDL input, prepass and compression statistics, and error messages.</p> <p>DCB=(LRECL=121,RECFM=FB,...)</p>
SYSUDUMP	O	Print data set (normally SYSOUT).

IMSPASS Parameters

The IMSPASS Parameter File (PARMFL) is an 80-character input record with:

1. Keyword parameters separated by commas appearing in columns 1-71.
2. More than one record may be input. Continuation is indicated by a comma following the last parameter on the record.
3. The last record is recognized by either the presence of data in column 71 (except for a comma), or no comma following the last parameter of the card. (The BEGIN parameter is an exception—see Table 8-2.)

Table 8-2. IMSPASS Parameter Specifications

Parameter Coding	Description
SEG=segname	Required. Specify name of the segment; must correspond to SEGM macro "segname" specification.
P= nnn	Required. Specify number of records (in thousands) to be prepassed. ALL specifies the entire data base. (P=003 specifies prepass 3000 records or the entire data base, whichever occurs first).
C= nnn	Optional. Specify number of records (in thousands) to be subjected to trial compression. ALL specifies the entire data base. (C=002 specifies prepass 2000 records or the entire data base, whichever occurs first). Usage is recommended to obtain compression statistics and verify RDL specifications beyond syntax check.
PCBNAME=dbdname or PCB=n (mutually exclusive)	Optional. Specify the name of the DBD from the PCB. Specify the relative number of the PCB in the PSB. If omitted, PCB=1 is assumed. Note: If the PSB was generated with CMPAT=YES, a "dummy PCB" is the first PCB in the PSB and must be accounted for when using PCB=n.
OUT= P or C	Optional. Specify number of records to be written to the OUTDB data set corresponding to number of records prepassed (OUT=P) or number of records "trial" compressed (OUT=C). If omitted, no records are written.
SKIP=nnnn	Optional. Specify number of input records to skip during IMSPASS execution to obtain a better cross-section of the segment type. (SKIP=1 specifies processing of records 2,4,6, etc.; SKIP=2 specifies processing of records 3,6,9, etc. Code as a decimal value from 1 to 9999.)
SKIPTO=nnnnn or, BEGIN=key (mutually exclusive)	Optional. SKIPTO specifies number of input records to skip prior to beginning prepass or(SKIPTO=1000 indicates start prepass at record 1001). Code as a decimal value from 1 to 99999. BEGIN specifies key at which prepass is to begin. If BEGIN=key is specified, - this parameter must follow the PCB or PCBNAME parameter. - the key specified must have the same length as the segment key. - code the key in character format (i.e., use multipunch for packed decimal representations). IMSPASS obtains the key length from IMS control blocks. If coding of the key requires use of column 71 of the input record, continuation is automatic. Do not code a comma to indicate continuation.

Sharing an FDT

To save CSA storage (5K to 24K per segment type), an FDT can be shared between segment types in two ways.

If the RDL statements associated with an FDT apply to several segment types and their data characteristics are similar, then their SEGCC macros may contain the parameter:

FN=fdtname

where fdtname is the name of the common FDT. This technique is often employed when a large data base has been partitioned into smaller, more manageable data bases, each with essentially the same segment types. For example, if there were seven compressed segment types in each of 20 similar data bases, we require only seven FDTs instead of 140.

The second way of sharing an FDT is to code the SEGCC parameter

FN=(fdtname,SHR)

where the RDL used to create fdtname need not correspond to the sharing segment type. In other words, the structure of the segments may be different (fixed vs. variable, different lengths, key lengths, etc.). The FDT must contain a type C1 field in its associated RDL. Only the type C1 data characteristics are extracted from the FDT, and these characteristics should approximately match the non-key portion of the sharing segment. (See the RDL chapter of this manual for a description of type C1 fields).

For example, if the FDT describes fields containing a mix of packed decimal and character data, the current segment should have more or less the same mix. If it were to contain binary data, expansion rather than compression might result.

A shared FDT need not correspond to any single segment type, but may be a merge of several segment types. Such an FDT can be constructed in the following manner:

1. Collect a representative data sampling from the various segments by running the IMSPASS utility for each, with

**OUT=P specified in PARMFL
DISP=(MOD,KEEP) on the OUTDB DD statement**

2. Run the File Prepass Utility using the OUTDB data set from step 1 as the input data set INFILE. Specify the RDL statement, Nx,CIVER, for the RECDEF, where x is the approximate average sequence field length. The average rather than the maximum sequence field length can be used, because for shared FDTs the RDL is not used to determine where compression starts.
3. Run the FDTLOADR Utility.
4. Unload the data base(s) with the old DBD and old FDTs if any.
5. For each segment type, code the SEGCC parameter

FN=(fdtname,SHR).

6. Reload the data base(s) with the new DBD and FDT.

```

// * IMSPASS JOB
// *
// *
// *****
// *
// * THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS for IMS USERS TO *
// * BUILD THE FILE DESCRIPTOR TABLE (FDT) *
// * *
// *****
// IMSPASS EXEC PGM=DFSRR00,PARM='DLI,IMSPASS,...'
// STEPLIB DD DISP=SHR,DSN=user.shrink.load {1}
// other private libraries, IMS load libraries,
// and standard IMS execution DD statements
// DFSRESLB DD DSN=IMSVS.RESLIB,DISP=SHR
// PRINT DD SYSOUT=*
// SYSUDUMP DD SYSOUT=*
// TABL00 DD DISP=(,CATLG,DELETE),DSN=user.fdt.file,
// UNIT=unit,VOL=SER=volser,SPACE=(TRK,(2,1))
// RECDEF DD *
// record definitions {2}
// *
// OUTDB DD DISP=(,CATLG,DELETE),DSN=user.outdb.file,
// UNIT=unit,VOL=SER=volser,SPACE=(TRK,(n,n))
// OUTFILE DD DUMMY,DCB=(LRECL=nn,BLKSIZE=nn)
// ddname DD DISP=SHR,DSN=user.dbd {3}
// PARMFL DD *
// SEG=segname,P=nnn,..... {4}
// *
// *

```

Figure 8-5. IMSPASS Execution JCL

Notes for Figure 8-5:

1. The DSN assigned to SHRINK.LOAD. Refer to the LOADPDS step of the INSTALL1 job from the Installation chapter.
2. RDL specification. See Figure 8-3.
3. The DDNAME and DSN of the DBD library for the specified segment.
4. The IMSPASS parameter statement. See Table 8-2.

Lower-case letters indicate user-supplied specifications in JCL.

Using the OUTDB Data Set

When you need to refine the RDL specification, you can experiment with revised RDL specifications. In addition to executing IMSPASS again with the new RDL and evaluating compression statistics, you can also create a sequential data set with test segments using OUTDB.

The OUTDB sequential data set becomes the INFILE data set to the File Prepass Utility. Descriptions for the P and C values of the PARM field are the same as described for the PARMFL DD statement of the IMSPASS Utility. After determining the proper RDL to use, execute IMSPASS again to create a new FDT. Do not

use the TABL00 file from the File Prepass Utility. For this reason, TABL00 is defined as a temporary data set.

This second option saves the overhead of repeated accesses of the data base. If you use this option, code the OUTDB DD statement and the OUT parameter of the IMSPASS PARMFL. When executing the File Prepass Utility, use the JCL shown in Figure 8-6.

```

/*IMSPAS2 JOB
/*
/*
/******
/*
/* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS FOR IMS USERS TO *
/* REVISE THEIR RDL SPECIFICATION *
/*
/******
//PREPASS EXEC PGM=SHRINK,PARM='P=ALL,C=ALL'
//STEPLIB DD DISP=SHR,DSN=user.shrink.load {1}
//PRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//INFILE DD DISP=OLD,DSN=user.outdb.file
//OUTFILE DD DUMMY
//TABL00 DD DISP(,PASS),DSN=&&FDTFILE,
// UNIT=unit,SPACE=(2048,(3,1))
//RECDEF DD *
//          revised RDL specification
/*
/*

```

Figure 8-6. File Prepass Utility JCL for RDL Revision

Note for Figure 8-6:

1. The DSN assigned to SHRINK.LOAD of the install tape. Refer to the LOADPDS step of the INSTALL1 job.

Executing the FDTLOADR Utility

The FDTLOADR Utility converts the File Descriptor Table written from IMSPASS to a load module. (For a more detailed description of the FDTLOADR Utility, see the SAMS:Compress/2 chapter of this manual.) The IMSPASS TABL00 data set corresponds to the TABL00 data set of the FDTLOADR Utility. The File Descriptor Table load modules must be stored in a special FDT load module library (SYSLMOD), which is required when executing SAMS:Compress for IMS. The module name “fdtname” must match the fdname defined in the FN field of the SEGCC macro. Note that in MVS, this load module is loaded into CSA (Common Storage Area). The names assigned to FDTs must be unique among all possible occupants of CSA.

The FDTLOADR Utility should be executed after the final IMSPASS execution, using the JCL in Figure 8-7.

```

/* IMSFLDR JOB
/*
/*
/******
/*
/* THE FOLLOWING JCL IS PROVIDED FOR SAMS:COMPRESS FOR IMS USERS TO *
/* CONVERT A SEQUENTIAL FDT TO LOAD MODULE FORMAT *
/*
/******
//CONVFDT EXEC PGM=FDTLOADR
//STEPLIB DD DISP=SHR,DSN=user.shrink.load {1}
//SYSLIN DD UNIT=SYSDA,SPACE=(TRK,(2,1))
//SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSLMOD DD DISP=SHR,DSN=user.fdt.load(fdtname)
//TABL00 DD DISP=(OLD,DELETE,KEEP),DSN=user.fdt.file

```

Figure 8-7. FDTLOADR Utility JCL

Note for Figure 8-7:

1. The DSN assigned to SHRINK.LOAD of the install tape. Refer to the LOADPDS step of the INSTALL1 job.

Performing Data Base Compression

Procedural Overview

For each data base to be compressed, the user must:

1. Unload the data base using the IMS reorganization/unload utility, DFSURGU0, using the current DBD.
2. Perform DBD generation, using an expanded SEGM macro (indicating a compression routine in use) and a SAMS:Compress for IMS SEGCC macro (indicating SAMS:Compress parameters) for each segment type to be compressed.
3. Reload the data base using the IMS reorganization/reload utility, DFSURGL0, and the new DBD. IMSHRINK is invoked as the compression exit to produce compressed segments. Figure 8-8 illustrates IMSHRINK operation.

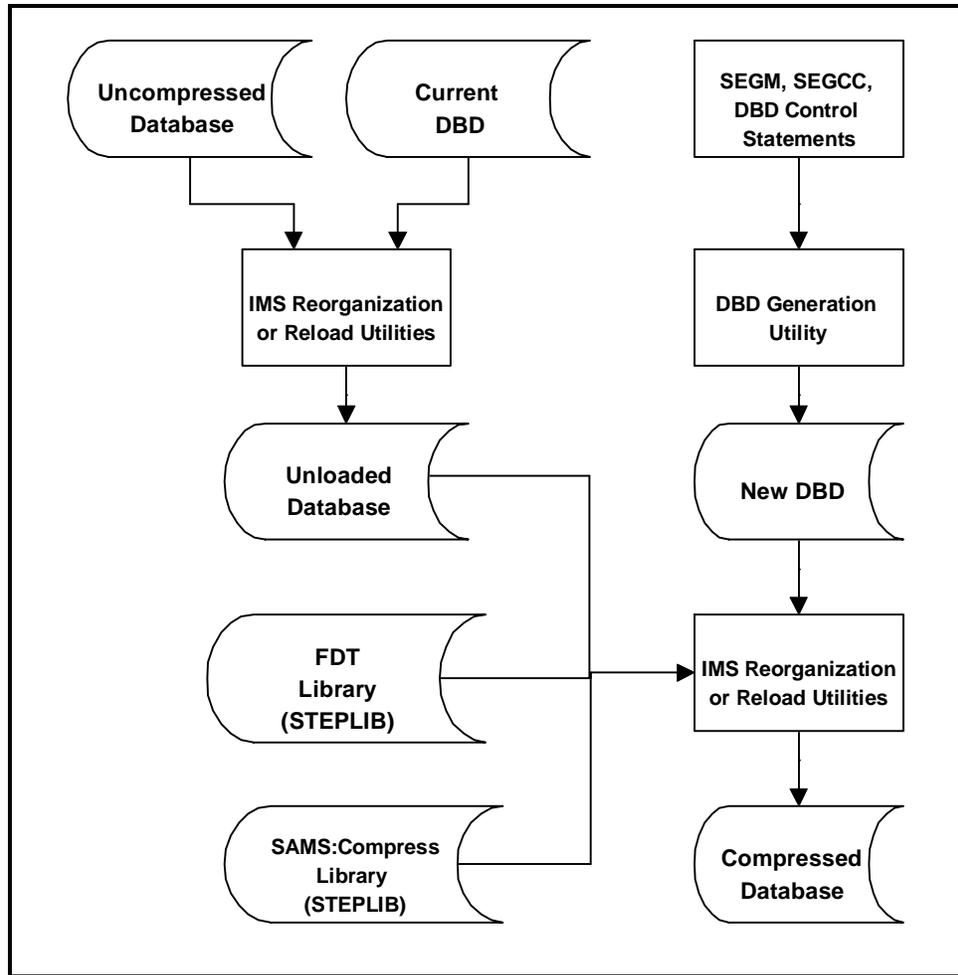


Figure 8-8. IMSHRINK Operation

Data Base Reorganization/Unload

Standard IMS procedures are used to unload the data base to be compressed. The IMS/VS reorganization/unload, DFSURGU0, utility is used, referencing the current (original) DBD with no COMPRTN defined.

DBD Generation

Each segment type to be compressed must be defined with an IMS SEGM macro coded with the COMPRTN parameter, and a SAMS:Compress for IMS SEGCC macro. Figure 8-9 illustrates DBD control statements.

Coding the SEGM Macro

The SEGM macro defining each segment type to be compressed must be coded with the COMPRTN parameter exactly as shown:

```
(symbol) SEGM      NAME=segname,COMPRTN=(IMSHRINK,DATA,INIT),
                  BYTES=[(max,min)],...
```

If a segment is variable length, the BYTES parameter must be altered as follows:

1. Add 12 to the max value.
2. Set the min value to the length of the sequence field plus 4.

Coding the SEGCC Macro

One SEGCC macro must be coded for each SEGM macro which specifies IMSHRINK as the compression routine. Coding format for the SEGCC macro is:

```
Segname
SEGCC      FN=fdtname      ,MINSIZE=n,STAT=YES,PAD=(N,xxx)
           (fdtname,SHR)   0          NO      (P,xx)
```

Segname corresponds to the NAME=segname parameter of a SEGM macro.

FN is an optional parameter. It is used to override the default FDT load module name to be associated with this segment. If omitted, the FDT load module name is assumed to be the same as the segname. When the default name is unacceptable due to name conflicts, FN must be coded. Code the SHR subparameter if this segment is to use an FDT created for a different segment type by the IMSPASS utility. In this case, it is not necessary to run IMSPASS for this current segment.

MINSIZE specifies the minimum length of a compressed segment. While not required, MINSIZE may be specified to reduce the probability of VSAM control interval splits during replacement operations. A MINSIZE specification can reduce the variation in length of compressed segments; during replacement operations, the segments are less likely to increase in length. A starting value for MINSIZE can be chosen from the trial compression statistics: a value between the minimum and average compressed segment length could be used. MINSIZE may be changed at any time without reloading the data base. If omitted, MINSIZE defaults to zero; i.e., no lower limit is imposed on the length of compressed segments. If a MINSIZE other than 0 is specified, it should be larger than the number of bytes from the beginning of the segment to the end of the segment key plus 5 bytes.

STAT specifies whether or not compression statistics are to be written to the system output writer at the time of the first IMS "close" for each segment type. The default value is YES. Code NO if statistics are not desired.

PAD specifies how much padding should be added to the end of a segment on segment insert only. This can be used to help reduce VSAM CI splits.

N,xxx = fixed Number of bytes, 1 to 255.

P,xx = Percentage, 1 to 99, calculated as $A + (((B-A)*P)/100)$,

where:

B = uncompressed segment length

A = compressed segment length

P = percentage specified by user

If omitted, no padding will be added.

```
DBD  NAME=filename,ACCESS=(VSAM,...)
      .
      SEGM NAME=SEGX,COMPRTN=(IMSHRINK,DATA,INIT),...
      SEGM NAME=SEGY,COMPRTN=(IMSHRINK,DATA,INIT),...
      .
      .
      .
      (Additional SEGM macros.  Not all segments need be compressed.)
      .
      DBDGEN
      .
      .
      .
      FINISH
      SEGX          SEGCC STAT=NO          {1}
      SEGY          SEGCC FN=FDT2         {2}
      .
      (Additional SEGCC macros)
      .
      END
```

Figure 8-9. Sample DBD Control Statements

Notes for Figure 8-9:

1. FDT name is same as segment name.
2. FDT name is given explicitly.

DBD Control Statement Assembly

Standard IMS procedures are used to assemble the DBD control statements. Ensure that the library containing the SAMS:Compress for IMS SEGCC macro is referenced at assembly time.

Data Base Reload

Execution of the data base reorganization/reload utility, DFSURGL0, follows standard IMS procedures with additional DD statements. The STEPLIB that contains the IMSHRINK and FDT load modules must be in the concatenation. See the “Installation” chapter for installation procedures for IMSHRINK.

Chapter 9. Implementing Express Without the Interactive Dialog

This chapter provides instructions to SAMS:Compress Express users who do not want to use the facilities of the Interactive Dialog (a supplied ISPF interface) to implement SAMS:Compress for IMS Express.

The Installation chapter of this manual provides step-by-step instructions for installing the SAMS:Compress family of products. If you are not using the Interactive Dialog, skip those sections dealing with its installation. However, you must follow all of the procedures for installing the SAMS:Compress for IMS Express product.

The main SAMS:Compress for IMS Express chapter of this manual provides a detailed product description and contains other reference materials which are applicable regardless of the method used to implement the product. Users should become familiar with the information presented there before trying to implement SAMS:Compress for IMS Express manually.

Once SAMS:Compress for IMS Express has been implemented, it is completely transparent to IMS applications programs. The applications programs require only minor JCL additions (i.e., STEPLIB DD statements to SAMS:Compress for IMS Express modules) to access compressed data base segments.

Implementation Overview

Because SAMS:Compress for IMS Express does not use FDTs, it does not have a SEGCC macro. Therefore, a user desiring to implement SAMS:Compress for IMS Express without using the Interactive Dialog must do so by modifying the DBD.

A segment is compressed when the following parameter is added to the segment macro that defines a segment to be compressed.

```
,COMPRTN=(IMEXPRES,DATA,INIT)
```

This parameter can be added by either of two methods:

Method I: Editing Existing DBD Source Macros

A user can activate the compression routine for a segment by editing the DBD source macro and adding the ,COMPRTN= parameter to each SEGM which is to be compressed.

Method II: Using a Data Dictionary to Generate DBD Source Macros

A user who utilizes a data dictionary to generate DBD source macros, should follow the procedure his dictionary calls for to modify a segment definition. In addition, he should add the ,COMPRTN= name (IMEXPRES) to the segment definition following the appropriate parameter. Once he has updated the data dictionary, he will then generate the new DBD source macros for the desired segments from the data dictionary.

After a user has obtained the new DBD source macros, he must assemble them and reorganize (REORG) the data base. During the REORG, a user must unload the data base using the current DBD macro, but he must load the data base using the new DBD macro.

Chapter 10. Reports

This chapter contains samples of the SAMS:Compress reports, along with descriptions of the sections and fields that appear on them. The following reports are described:

Reports Common to all SAMS:Compress Products

- FDT Index Listing
- Authorized User List

SAMS:Compress for IMS Reports

- Index List
- Stored Data Report
- Extracted Data Sample Report
- Byte Distribution Analysis Report
- Test Compression Report

Reports Common to All SAMS:Compress Products

FDT Index Listing

This report lists all of the FDTs currently contained in the Interactive Dialog Control file.

LIST OF FDTs IN THE SAMS:COMPRESS FOR IMS INTERACTIVE DIALOG CONTROL FILE AS OF 01/25/94 AT 15:50										PAGE: 1
STERLING SOFTWARE, INC.										
INTER R <----- S T R U C T U R E ----->					<----- FDT CREATION DATA ----->					
FDT ID	FDT NAME	FACE	B NAME	TITLE	CREATED BY	DATE	TIME	REL		
FDT00054	SEGMENTA	IMS	G SAMPLEDB	TEST MMM 523	MARY MARIE MELLMANN	01/11/94	11:41	5.2.3		
FDT00055	SEGMENTB	IMS	G SAMPLEDB	TEST MMM 523	MARY MARIE MELLMANN	01/11/94	11:41	5.2.3		
FDT00056	SEGMENTC	IMS	G SAMPLEDB	TEST MMM 523	MARY MARIE MELLMANN	01/11/94	11:41	5.2.3		
FDT00057	SEGMENTE	IMS	G SAMPLEDB	TEST MMM 523	MARY MARIE MELLMANN	01/11/94	11:41	5.2.3		
FDT00058	SEGMENTF	IMS	G SAMPLEDB	TEST MMM 523	MARY MARIE MELLMANN	01/11/94	11:41	5.2.3		

Figure 10-1. FDT Index Listing

company name—The name of the company as entered during the install process.

FDT ID—The control ID for the FDT. The numeric portion of the ID is the FDT number, which is used in any error message.

FDT NAME—The load module name for the FDT.

INTERFACE—The interface (IMS or MVS) for which the FDT was generated.

RB—The Recorded By (RB) column. A “G” means the FDT was generated by the Interactive Dialog.

STRUCTURE—Information about the file or database.

NAME—The data set name or the DBD and segment names.

TITLE—The title of the entry.

FDT CREATION DATA—Information about the person who generated the file or database, when it was generated, and the release used.

CREATED BY—The name of the person who generated the FDT.

DATE—The date the FDT was generated.

TIME—The time the FDT was generated.

REL—The release of the Interactive Dialog that was used to generate the FDT.

Authorized User List

This report lists all persons authorized to use the Interactive Dialog Control File.

SAMS:COMPRESS DIALOG AUTHORIZED USER LIST AS OF: DATE: 01/25/94 TIME: 15:49 PAGE: 1								
STERLING SOFTWARE, INC.								
			----- SAMS:COMPRESS DIALOG USAGE -----					
TSO ID	USER'S FULL NAME	NICK NAME	IMS	MVS	IDMS	DB2	FDT	SEC
-----	-----	-----	-----	-----	-----	-----	-----	-----
ISPAPT1	THOMAS, ANDREA	ANDREA	YES	NO	NO	NO	YES	YES
ISPDNP1	PHILLIPS, DEVIN	DEVIN	YES	NO	NO	NO	NO	NO
ISPAMM1	MCPHEARSON, ADAM	ADAM	YES	NO	NO	NO	NO	NO
ISPBW1	WILLIAMS, BARBARA	BARBARA	YES	NO	NO	NO	NO	NO
ISPKTS1	SHORT, KENETH	KEN	YES	NO	NO	NO	NO	NO
ISPELM1	MARSHALL, ELAINE	ELAINE	YES	NO	NO	NO	NO	NO
ISPTXF1	FERRIS, TOM	TOM	YES	NO	NO	NO	NO	NO
ISPWAS1	SMITH, WILLIAM	BILL	YES	NO	NO	NO	NO	NO
ISPSAA1	ADAMS, SHEILA	SHEILA	YES	NO	NO	NO	NO	NO

Figure 10-2. Authorized User List

company name—The name of the company as entered during the install process.

TSO ID—This is the TSO ID of the user. Since the Interactive Dialog is an ISPF dialog, it uses the TSO ID as the primary user identifier.

USER'S FULL NAME—This is the name of the user that is recorded on the Interactive Dialog Control File to identify the user who performed certain maintenance functions or services.

NICK NAME—This is the name the user wishes to have appear in the various HELP and TUTORIAL messages and panels in the field where the user's name is displayed.

SAMS:Compress DIALOG USAGE—This section indicates which of the various Dialog services the user is to be permitted to use.

IMS—SAMS:Compress for IMS services.

MVS—SAMS:Compress for MVS services.

IDMS—SAMS:Compress for IDMS (not supported).

DB2—SAMS:Compress for DB2 services (not supported).

FDT—FDT re-generate and delete services.

SEC—Modify installation control parameters, perform user maintenance, and set default data set names.

IMS Reports

Index List

This report lists all of the databases currently defined to the Interactive Dialog Control File.

LIST OF IMS DATA BASES IN THE SAMS:COMPRESS INTERACTIVE DIALOG CONTROL FILE AS OF 01/25/94 AT 15:51										PAGE: 1
STERLING SOFTWARE, INC.										
DBD NAME	DATA BASE NAME	<---- IMS DATA ---->			<--- SAMS:COMPRESS FOR IMS DATA ----->					
-----	-----	GEN DATE	TIME	REL	ANALYZED BY	DATE	TIME	REL	-----	
SAMPLEDB	MMM TEST 523	01/05/94	10.29	3.1.0	MARY MARIE MELLMANN	01/11/94	11:44	5.2.3		
TRAINDB	TEST	01/22/93	15.01	3.1.0	MARY MARIE MELLMANN	01/22/93	15:19	5.2.B		

Figure 10-3. Index List

company name—The name of the company as entered during the install process.

DBD NAME—The name of the DBD.

DATA BASE NAME—The name of the database.

IMS DATA—Information about when the DBD load module and the IMS release used.

GEN DATE—The date the DBD load module was assembled.

TIME—The time the DBD load module was assembled.

REL—The release of the IMS used to assemble the DBD.

SAMS:Compress for IMS DATA—Information about the person that analyzed the database, the date and time it was analyzed, and the release used.

ANALYZED BY—The name of the person that performed the database structure analysis.

DATE—The date the database structure was analyzed.

TIME—The time the database structure was analyzed.

REL—The release of the Interactive Dialog that was used to analyze the database.

Stored Data Report

The IMS Stored Data report for a database displays most of the information that has been recorded about an IMS database. The report consists of four basic parts.

1. Page Heading and Database Information section.
2. Segment section.
3. Data Set Information section.
4. JCL Generation Information section.

Page Heading and Database Information Section

This section pertains to the database in general.

```

STORED DATA REPORT FOR IMS DBD -- HIDAMDB -- HIDAM DATABASE
STERLING SOFTWARE, INC.                               AS OF 01/25/94 AT 16:03                PAGE: 1
DATA BASE INFORMATION:*****
* ANALYZED ON 06/03/93 AT 13:58 BY MARY MARIE MELLMANN FROM SHRINK.IMSEXP.DBDLIB *
* DBD WAS ASSEMBLED ON 06/03/93 AT 11.52 WITH RELEASE 3.1.0 OF IMS *
* ORGANIZATION IS HIDAM DATA BASE CONTAINS: *
*           1 DATA SET GROUPS *
*           3 LEVELS *
*           3 SEGMENTS *
*           0 2ND INDEXES *
*           1 ASSOCIATED DBDS -- HIDAMNDX *
* PRODUCTION DBD WAS GENERATED ON 06/03/93 AT 14:06 BY MARY MARIE MELLMANN *
*****

```

Figure 10-4. Page Heading and Database Information Section

company name—The name of the company as entered during the install process.

ANALYZED—SAMS:Compress for IMS analysis information.

ON—The date database structure was analyzed.

AT—The time the database structure was analyzed.

BY—The name of the person that performed the analysis.

FROM—Data set name of the DBD library from which analysis was performed.

DBD WAS ASSEMBLED—Information about the assembly of the DBD.

ON—The date and time the database DBD was assembled.

AT—The date and time the database DBD was assembled.

WITH RELEASE—The release of IMS used to perform the assembly.

ORGANIZATION IS—The database organization.

DATA BASE CONTAINS—The contents of the database.

DATA SET GROUPS—The number of Data Set Groups (DSG) in the database.

LEVELS—The number of levels in the database.

SEGMENTS—The number of segments in the database.

2ND INDEXES—The number of secondary indexes that point to the database.

ASSOCIATED DBDS—The number of other DBDs that are associated with this database. This will include the primary and secondary indexes, any logically related databases and their indexes, etc., and the DBD names of all the associated databases.

PRODUCTION DBD WAS GENERATED—The SAMS:Compress DBD generation information, if the DBD has been processed through the IMS implementation process.

ON—The date DBD was generated.

AT—The time the DBD was generated.

BY—The name of the person that performed the generate.

Segment Section

The Segment section contains information about a specific segment, which consists of a root segment and a dependent segment. The two segment reports are nearly identical, except that they apply to the parent and child respectively.

Root Segment

```

SEGMENT 1 IS CTROOTSG ON LEVEL 1 AND IS A CHILD OF *****
* FORMAT(F) MIN LENG( 0) MAX LENG( 40) PFX LENG( 10) SECONDARY INDEX SOURCE(NO ) *
* KEY FIELD(ROOTKEY ) STARTS AT( 1) LENGTH( 7) OCCURRENCES( 3) *
* MAXIMUM COMPRESSIBLE LENGTH( 33) THE CURRENT RDL SOURCE STATEMENT IS: *
* *
* |N7, |
* | CLF33. |
* *
* "FDT" NAME(CTROOTSG) SHARE(NO ) MINIMUM COMPRESSED SIZE( 0) COMPRESS STATS(YES) PADDING -- LENGTH( 0) TYPE(FIXED ) *
* "FDT" IS NOT SHARED *
* SEGMENT WAS SELECTED FOR EXPRESS COMPRESSION ON 06/03/93 AT 14:05 BY MARY MARIE MELLMANN *
* DATA EXTRACT PARAMETERS ARE -- BYPASS( 0) EXTRACT (ALL) SKIP( 0) *
* DATA SAMPLE CONSISTS OF 3 SEGMENTS EXTRACTED ON 06/03/93 AT 14:04 *
* INPUT LENGTHS: SMALLEST = 40 AVERAGE = 40 LARGEST = 40 *
* SAMPLE LENGTHS: SMALLEST = 40 AVERAGE = 40 LARGEST = 40 *
* BYTE DISTRIBUTION ANALYSIS WAS COMPLETED ON 06/03/93 AT 14:04 *
* THE LAST COMPRESSION TEST WAS RUN ON 06/03/93 AT 14:04 *
* THE BEST COMPRESSION WAS ACHIEVED BY TEST NBR -- 1 WHEN THE AVERAGE COMPRESSED SEGMENT SIZE WAS 46 BYTES *
* BEST TEST LENGTHS: SMALLEST = 35 AVERAGE = 36 LARGEST = 36 *
* LAST HUFF LENGTHS: SMALLEST = 35 AVERAGE = 36 LARGEST = 36 *
* LAST EXPR LENGTHS: SMALLEST = 40 AVERAGE = 40 LARGEST = 40 *
*****

```

Figure 10-5. Root Segment

SEGMENT—Hierarchical position of the segment within the database.

IS—The name of the segment.

ON LEVEL—The level of the database that contains the segment.

AND IS A CHILD OF—This field is blank for the root segment.

FORMAT—The format of the segment (Fixed or Variable).

MIN LENG—The minimum length of the segment (0 if fixed).

MAX LENG—The maximum length of the segment.

PFX LENG—The length of the IMS prefix for the segment.

SECONDARY INDEX SOURCE—An indicator as to whether the segment is the source of a secondary index.

KEY FIELD—The name of the key field of the segment. This field will be blank for non-keyed segments.

STARTS AT—The starting position of the segment key. This field will be 0 for non-keyed segments.

LENGTH—The length of the segment key. This field will be 0 for non-keyed segments.

OCCURRENCES—The number of occurrences of the segment within the database. This number is based upon the average number of this segment per root segment read during the extraction of the data sample. The user must supply the number of root segments in the database after the data sample has been extracted but before trying to test compress the data sample for the database.

MAXIMUM COMPRESSIBLE LENGTH—The number of compressible bytes of data in the segment.

THE CURRENT RDL SOURCE STATEMENT IS—The current RDL for the segment.

After the implementation process has been performed, a message will appear indicating that the FDT has been generated if full service compression was selected or if Express compression was selected, the message will indicate the use of Express.

"FDT" NAME—The current FDT parameters for the segment.

SHARE—Indicates whether or not the FDT is shared.

MINIMUM COMPRESSED SIZE—The value specified for MINSIZE, which is the minimum size of the compressed segment.

COMPRESS STATS—Indicates whether or not compression statistics are to be generated.

PADDING—The length and type of padding used.

LENGTH—The length of padding used.

TYPE—The type of padding used.

"FDT" IS NOT SHARED—Indicates whether or not the FDT is shared.

SEGMENT WAS SELECTED—Information about the selection of the segment.

FOR—The type of compression used for the segment.

ON—The date the segment was selected.

AT—The time the segment was selected.

BY—The name of the person that performed the selection.

DATA EXTRACTION PARAMETERS ARE—The data extraction parameters used to create the sample. This line will be omitted from the dependent segments.

BYPASS—The number of records to be BYPASSed prior to extracting the first record. This number is a multiple of 1,000 records.

EXTRACT—The number of records that are to be extracted. If ALL appears, it means that once the extraction begins, it is to continue to the end of the database. If a number is shown, it means that this many thousands of records are to be extracted.

SKIP—The number of records to be SKIPPed between each extracted record. It spreads the extraction across the database. A value of 0 means that once extraction has begun, every record is to be extracted. A value of 3 means that between each extracted record, 3 records will be skipped. This will yield a data sample that is 25% of the database.

DATA SAMPLE CONSISTS OF—After the data sample has been extracted, the number of segments in the sample.

ON—The date the sample was extracted.

AT—The time the sample was extracted.

INPUT LENGTHS—The lengths of the smallest, the average, and the largest segments read.

SMALLEST—The length of the smallest segment read.

AVERAGE—The length of the average segment read.

LARGEST—The length of the largest segment read.

SAMPLE LENGTHS—The lengths of the smallest, the average, and the largest segments extracted.

SMALLEST—The length of the smallest segment extracted.

AVERAGE—The length of the average segment extracted.

LARGEST—The length of the largest segment extracted.

BYTE DISTRIBUTION ANALYSIS WAS COMPLETED—After the Byte Distribution has been performed, the data and time that the Byte Distribution was run. This field is blank if the Byte Distribution Analysis has not yet been run.

ON—The date the Byte Distribution Analysis was completed.

AT—The time the Byte Distribution Analysis was completed.

LAST COMPRESSION TEST WAS RUN—The date and time of the most recent compression test.

ON—The date of the most recent compression test.

AT—The time of the most recent compression test.

BEST COMPRESSION WAS ACHIEVED BY TEST NBR—The number of the test that obtained the best compression for the segment. This field is blank if the Byte Distribution Analysis has not yet been run.

WHEN THE AVERAGE COMPRESSED SEGMENT SIZE WAS—The average size in bytes of the segment that achieved the best compression.

BEST TEST LENGTHS—The lengths that obtained the best test compression for the segment.

SMALLEST—The length of the smallest, best compression test.

AVERAGE—The length of the average, best compression test.

LARGEST—The length of the largest, best compression test.

LAST HUFF LENGTHS—The lengths of the most recent Huffman compression test for the segment.

SMALLEST—The length of the smallest Huffman compression test.

AVERAGE—The length of the average Huffman compression test.

LARGEST—The length of the largest Huffman compression test.

LAST EXPR LENGTHS—The lengths of the most recent Express compression test for the segment.

SMALLEST—The length of the smallest Express compression test.

AVERAGE—The length of the average Express compression test.

LARGEST—The length of the largest Express compression test.

Dependent Segment

```

SEGMENT 2 IS CTPRTRSG ON LEVEL 2 AND IS A CHILD OF CTROOTSG *****
* FORMAT(F) MIN LENG( 0) MAX LENG( 150) PFX LENG( 14) SECONDARY INDEX SOURCE(NO )
* KEY FIELD(LEVEL2KF) STARTS AT( 1) LENGTH( 16) OCCURRENCES( 9)
* MAXIMUM COMPRESSIBLE LENGTH( 134) THE CURRENT RDL SOURCE STATEMENT IS:
*
* |N16, |
* | C1F134. |
*
* "FDT" NAME(CTPRTRSG) SHARE(NO ) MINIMUM COMPRESSED SIZE( 0) COMPRESS STATS(YES) PADDING -- LENGTH( 0) TYPE(FIXED )
* "FDT" IS NOT SHARED
* "FDT" GENERATED ON 06/03/93 AT 14:05 BY MARY MARIE MELLMANN
* DATA SAMPLE CONSISTS OF 9 SEGMENTS EXTRACTED ON 06/03/93 AT 14:04
* INPUT LENGTHS: SMALLEST = 150 AVERAGE = 150 LARGEST = 150
* SAMPLE LENGTHS: SMALLEST = 150 AVERAGE = 150 LARGEST = 150
* BYTE DISTRIBUTION ANALYSIS WAS COMPLETED ON 06/03/93 AT 14:04
* THE LAST COMPRESSION TEST WAS RUN ON 06/03/93 AT 14:04
* THE BEST COMPRESSION WAS ACHIEVED BY TEST NBR -- 1 WHEN THE AVERAGE COMPRESSED SEGMENT SIZE WAS 83 BYTES
* BEST TEST LENGTHS: SMALLEST = 59 AVERAGE = 69 LARGEST = 72
* LAST HUFF LENGTHS: SMALLEST = 59 AVERAGE = 69 LARGEST = 72
* LAST EXPR LENGTHS: SMALLEST = 95 AVERAGE = 116 LARGEST = 136
*****

```

Figure 10-6. Dependent Segment

SEGMENT—Hierarchical position of the segment within the database.

IS—The name of the segment.

ON LEVEL—The level of the database that contains the segment.

AND IS A CHILD OF—The name of the physical parent of the segment. This area is blank for the root segment.

FORMAT—The format of the segment (Fixed or Variable).

MIN LENG—The minimum length of the segment (0 if fixed).

MAX LENG—The maximum length of the segment.

PFX LENG—The length of the IMS prefix for the segment.

SECONDARY INDEX SOURCE—An indicator as to whether the segment is the source of a secondary index.

KEY FIELD—The name of the key field of the segment. This field will be blank for non-keyed segments.

STARTS AT—The starting position of the segment key. This field will be 0 for non-keyed segments.

LENGTH—The length of the segment key. This field will be 0 for non-keyed segments.

OCCURRENCES—The number of occurrences of the segment within the database. This number is based upon the average number of this segment per root segment read during the extraction of the data sample. The user must supply the number of root segments in the database after the data sample has been extracted but before trying to test compress the data sample for the database.

MAXIMUM COMPRESSIBLE LENGTH—The number of compressible bytes of data in the segment.

THE CURRENT RDL SOURCE STATEMENT IS—The current RDL for the segment.

After the implementation process has been performed, a message will appear indicating that the FDT has been generated if full service compression was selected or if Express compression was selected, the message will indicate the use of Express.

"FDT" NAME—The current FDT parameters for the segment.

SHARE—Indicates whether or not the FDT is shared.

MINIMUM COMPRESSED SIZE—The value specified for MINSIZE, which is the minimum size of the compressed segment.

COMPRESS STATS—Indicates whether or not compression statistics are generated.

PADDING—The length and type of padding used.

LENGTH—The length of padding used.

TYPE—The type of padding used.

"FDT" IS NOT SHARED—Indicates whether or not the FDT is shared.

"FDT" GENERATED—Information about the generation of the FDT.

ON—The date the FDT was generated.

AT—The time the FDT was generated.

BY—The name of the person that performed the generate.

The root segment will contain the Data Sample Extract parameters. This line will be omitted from the dependent segments.

DATA SAMPLE CONSISTS OF—After the data sample has been extracted, the number of segments in the sample will be shown along with the date and time the sample was extracted.

SEGMENTS EXTRACTED—The date and time that the sample was extracted.

ON—The date the sample was extracted.

AT—The time the sample was extracted.

INPUT LENGTHS—The lengths of the smallest, the average, and the largest segments read.

SMALLEST—The length of the smallest segment read.

AVERAGE—The length of the average segment read.

LARGEST—The length of the largest segment read.

SAMPLE LENGTHS—The lengths of the smallest, the average, and the largest segments extracted.

SMALLEST—The length of the smallest segment extracted.

AVERAGE—The length of the average segment extracted.

LARGEST—The length of the largest segment extracted.

BYTE DISTRIBUTION ANALYSIS WAS COMPLETED—After the Byte Distribution has been performed, the data and time that the Byte Distribution was run. This field is blank if the Byte Distribution Analysis has not yet been run.

ON—The date the Byte Distribution Analysis was completed.

AT—The time the Byte Distribution Analysis was completed.

LAST COMPRESSION TEST WAS RUN—The date and time of the most recent compression test.

ON—The date of the most recent compression test.

AT—The time of the most recent compression test.

BEST COMPRESSION WAS ACHIEVED BY TEST NBR—The number of the test that obtained the best compression for the segment. This field is blank if the Byte Distribution Analysis has not yet been run.

WHEN THE AVERAGE COMPRESSED SEGMENT SIZE WAS—The average size in bytes of the segment that achieved the best compression.

BEST TEST LENGTHS—The lengths that obtained the best test compression for the segment.

SMALLEST—The length of the smallest, best compression test.

AVERAGE—The length of the average, best compression test.

LARGEST—The length of the largest, best compression test.

LAST HUFF LENGTHS—The lengths of the most recent Huffman compression test for the segment.

SMALLEST—The length of the smallest Huffman compression test.

AVERAGE—The length of the average Huffman compression test.

LARGEST—The length of the largest Huffman compression test.

LAST EXPR LENGTHS—The lengths of the most recent Express compression test for the segment.

SMALLEST—The length of the smallest Express compression test.

AVERAGE—The length of the average Express compression test.

LARGEST—The length of the largest Express compression test.

Data Set Information Section

The Data Set Information section provides information about the data sets that are required for IMS to process the database.

```

DATA SET INFORMATION:*****
* DDNAME(HIDAMDD ) DEV(3380) DBD(HIDAMDB ) DSN(SHRINK.HIDAM.DATABASE ) TRACKS( 60) SEGS( 21) *
* 'DATA SET NAME' ENTERED ON 06/03/93 AT 13:58 BY MARY MARIE MELLMANN *
*   DSG   LENGTHS:  SMALLEST =   50 AVERAGE =  226 LARGEST =  345 *
*   SAMPLE LENGTHS:  SMALLEST =   50 AVERAGE =  226 LARGEST =  345 *
*   HUFFMAN LENGTHS:  SMALLEST =   45 AVERAGE =   91 LARGEST =  117 *
*   EXPRESS LENGTHS:  SMALLEST =   50 AVERAGE =  174 LARGEST =  253 *
* DDNAME(HIDAMNDX) DEV(3380) DBD(HIDAMNDX) DSN(SHRINK.HIDAM.DATABASE.INDEX ) TRACKS( 1) SEGS( 0) *
* 'DATA SET NAME' ENTERED ON 06/03/93 AT 13:58 BY MARY MARIE MELLMANN *
*****

```

Figure 10-7. Data Set Information Section

DDNAME—The DDNAME for the data set.

DEV—The type of device on which the data set is stored.

DBD—The name of the DBD in which the DDNAME was found.

DSN—The name of the data set.

TRACKS—The number of tracks allocated to the data set.

SEGS—The number of segments that are found in the data sets that correspond to the DSGs in the database.

'DATA SET NAME' ENTERED—The event information about who supplied the data set name and when.

ON—The date the data set name was entered.

AT—The time the data set name was entered.

BY—The name of the person that entered the data set name.

DSG LENGTHS—The length of the DSG segments.

SMALLEST—The length of the smallest segment from the DSG.

AVERAGE—The length of the average segment from the DSG.

LARGEST—The length of the largest segment from the DSG.

SAMPLE LENGTHS—The length of the sample segments.

SMALLEST—The length of the smallest segment from the sample.

AVERAGE—The length of the average segment from the sample.

LARGEST—The length of the largest segment from the sample.

HUFFMAN LENGTHS—The length of the segments using Huffman.

SMALLEST—The length of the smallest segment with Huffman.

AVERAGE—The length of the average segment with Huffman.

LARGEST—The length of the largest segment with Huffman.

EXPRESS LENGTHS—The length of the segments using Express.

SMALLEST—The length of the smallest segment with Express.

AVERAGE—The length of the average segment with Express.

LARGEST—The length of the largest segment with Express.

DDNAME—The DDNAME for the index.

DEV—The type of device on which the index is stored.

DBD—The name of the DBD in which the DDNAME for the index was found.

DSN—The name of the index.

TRACKS—The number of tracks allocated to the index.

SEGS—The number of segments that are found in the index that correspond to the DSGs in the database.

'DATA SET NAME' ENTERED—The event information about who supplied the index name and when.

ON—The date the index name was entered.

AT—The time the index name was entered.

BY—The name of the person that entered the index name.

JCL Generation Information Section

The JCL Generation Information section provide information about what has transpired during the compression testing for the database.

```

JCL GENERATION INFORMATION:*****
* SAMPLE DATA EXTRACTION JCL WAS GENERATED ON 06/03/93 AT 13:59 BY MARY MARIE MELLMANN      TO EXTRACT USING DLI CALLS IN BATCH *
* PSBLIB   = ISPMMML. IMS.PSBLIB                      PSB NAME = HIDAMPSB  PCB NUMBER =   1 *
* ACBLIB   = ISPMMML. IMS.PSBLIB                      PSB NAME = HIDAMPSB  PCB NUMBER =   1 *
* ACBLIB   = ISPMMML. IMS.PSBLIB                      PSB NAME = HIDAMPSB  PCB NUMBER =   1 *
* RESLIB   = IMSESA. IMSS.RESLIB                      PSB NAME = HIDAMPSB  PCB NUMBER =   1 *
* DBDLIB   = SHRINK. IMSEXP.DBDLIB                    *
* EXTRACT  = ISPMMML.HIDAM.EXTRACT                    *
* THE DATA SAMPLE FOR THE DATA BASE WAS EXTRACTED ON 06/03/93 AT 14:04 *
* BYTE DISTRIBUTION ANALYSIS JCL WAS GENERATED ON 06/03/93 AT 13:59 BY MARY MARIE MELLMANN *
* THE BYTE DISTRIBUTION ANALYSIS WAS LAST RUN ON 06/03/93 AT 14:04 *
* COMPRESSION TESTING JCL WAS GENERATED ON 06/03/93 AT 13:59 BY MARY MARIE MELLMANN *
* COMPRESSION TESTING FOR THE DATA BASE WAS LAST RUN ON 06/03/93 AT 14:04 FOR COMPRESSION TEST NUMBER 1. *
*****

```

Figure 10-8. JCL Generation Information Section

SAMPLE DATA EXTRACTION JCL WAS GENERATED—JCL extraction information.

ON—The date the extract JCL was generated.

AT—The time the extract JCL was generated.

BY—The name of the person who generated the extract JCL.

PSBLIB—The PSBLIB data set name used during the data sample extraction.

PSB NAME—The name of the PSB.

PCB NUMBER—The number of the PCB.

ACBLIB—The ACBLIB data set name used during the data sample extraction.

PSB NAME—The PSB name for the ACB.

PCB NUMBER—The PCB number for the ACB.

RESLIB—The RESLIB data set name used during the data sample extraction.

DBDLIB—The DBDLIB data set name used during the data sample extraction.

EXTRACT—The name of the data set to which the extracted data was written.

THE SAMPLE DATA FOR THE DATA BASE WAS EXTRACTED—The date and time that the sample data was extracted.

ON—The date the sample data was extracted.

AT—The time the sample data was extracted.

BYTE DISTRIBUTION ANALYSIS WAS GENERATED—Byte Distribution Analysis JCL Generation information.

ON—The date the byte distribution JCL was generated.

AT—The time the byte distribution JCL was generated.

BY—The name of the person who generated the byte distribution JCL.

BYTE DISTRIBUTION ANALYSIS WAS LAST RUN—The date and time of the most recent Byte Distribution Analysis JCL Generation.

ON—The date of the most recent byte distribution run.

AT—The time of the most recent byte distribution run.

COMPRESSION TESTING JCL WAS GENERATED—Compression testing JCL Generation information.

ON—The date the compression testing JCL was generated.

AT—The time the compression testing JCL was generated.

BY—The name of the person who generated the compression testing JCL.

COMPRESSION TESTING FOR THE DATA BASE WAS LAST RUN—The date and time, number of the most recent test compression run.

ON—The date of the most recent test compression run.

AT—The time of the most recent test compression run.

FOR COMPRESSION TEST NUMBER—The number of the last test compression run.

Extracted Data Sample Report

The Extracted Data Sample Report provides some basic information about the data sample used to base the compression of the segment. The report consists of a heading section and an extraction section.

Heading Section

The heading section identifies the database, the installation, and the date and the time the report was generated.

```

EXTRACTED DATA SAMPLE REPORT FOR IMS DATA BASE -- SAMPLEDB -- MMM TEST 523
STERLING SOFTWARE, INC. AS OF 01/26/94 AT 14:45 PAGE: 1
EXTRACT WAS FROM A HDAM DATA BASE USING PCB 1 IN PSB SAMPLPSB
EXTRACT DATA WRITTEN TO DATA SET SHRINK.EXTRACT.FILE

```

Figure 10-9. Heading Section

EXTRACTED SAMPLE REPORT FOR IMS DATA BASE—The title of the report, the name of the data set, and test number.

company name—The name of the company as entered during the install process.

AS OF—The date the report was generated.

AT—The time the report was generated.

EXTRACTED FROM A—The type of data set from which the data was extracted.

USING—The PCB used during the data extraction.

IN PSB—The PSB used during the data extraction.

EXTRACT DATA WRITTEN TO—The name of the data set that contains the extracted data sample.

Data Section

The data portion of the extract report will express the information in terms of database records and segments for IMS databases.

<----- SEGMENT DATA ----->						<- EXTRACT CNTLS ->			<----- DATA SAMPLE EXTRACT INFORMATION ----->			
HIER	SEGMENT	SEG	PFX	SEGMENT	LENGTH	BYPASS	COUNT	SKIP	SEGMENTS	SEGMENTS	SEGMENTS	BYTES
POS	NAME	FMT	LEN	MIN	MAX	X1000	X1000	X1	READ	BYPASSED	EXTRACTED	EXTRACTED
1	SEGMENTA	F	18		100	0	ALL	0	3	0	3	300
2	SEGMENTB	V	10	100	312				9	0	9	1,312
3	SEGMENTC	F	6		80				9	0	9	720
4	SEGMENTD	V	6	50	200				3	0	3	576
5	SEGMENTE	V	10	100	212				1	0	1	192
6	SEGMENTF	F	10		80				1	0	1	80
7	SEGMENTG	F	6		100				2	0	2	200
EXTRACT RUN TOTALS:									28	0	28	3,380
THE AVERAGE LENGTH OF THE RECORDS READ WAS						1,127 BYTES.						
THE AVERAGE LENGTH OF THE RECORDS EXTRACTED WAS						1,127 BYTES.						
THIS IS A VARIATION IN AVERAGE RECORD LENGTHS OF						0.0000 PERCENT.						
THE PERCENT OF VARIATION BETWEEN THE AVG DATABASE RECORD LENGTHS CAN CAUSE A LIKE ERROR IN THE DASD SPACE NEEDED FOR THE DATA BASE												

Figure 10-10. Data Section

SEGMENT DATA—The characteristics of the uncompressed data.

HIER POS—The hierarchical position of the segment within an IMS database.

SEGMENT NAME—The name of the segment in an IMS database.

SEG FMT—The format of the segment, Fixed or Variable.

PFX LEN—The length of the IMS segment prefix in bytes.

SEGMENT LENGTH MIN—Only applies to variable length IMS database segments and contains the minimum segment length.

SEGMENT LENGTH MAX—The maximum length of the segment.

EXTRACT CNTLS—The criteria for extracting the data sample. It is in terms of database records or root segments.

BYPASS—The number of records BYPASSED prior to extracting the first record. This number is a multiple of 1,000 records.

COUNT—The number of records extracted. If ALL appears, it means that once the extraction begins, it is to continue to the end of the database. If a number is shown, it means that this many thousands of records were extracted.

SKIP—The number of records SKIPPED between each extracted record. It spreads the extraction across the database. A value of 0 means that every record was extracted. A value of 3 means that between each extracted record, 3 records were skipped. This data sample is 25% of the database.

DATA SAMPLE EXTRACT INFORMATION—The number of segments processed during the data extraction. The reading of the database will terminate as soon as the desired number of records have been extracted.

SEGMENTS READ—The total number of segments read during the data extraction.

SEGMENTS BYPASSED—The number of segments that were BYPASSED and/or SKIPPED during the data extraction.

SEGMENTS EXTRACTED—The number of segments that were written to the extracted data sample file.

BYTES EXTRACTED—The number of bytes of data contained in the extracted data sample. If the average length of the segments that is read are not within .01 percent of the average length of the segments extracted, a second line will be printed showing the average lengths and the percent of variation.

EXTRACT RUN TOTALS—Totals for all segments for the preceding 4 extraction items.

THE AVERAGE LENGTH OF THE RECORDS READ WAS—The average length of all the records read.

THE AVERAGE LENGTH OF THE RECORDS EXTRACTED WAS—The average length of only the records extracted.

THIS IS A VARIATION IN AVERAGE RECORD LENGTHS OF—The variance between the average length of records read and records extracted.

Average record length information is displayed to aide in determining whether the extracted sample is representative of the database as a whole. If the lengths of the database records read and of the database records extracted are not close to each other, then the data extraction parameters should be modified and another data sample should be extracted. All of the DASD space and cost calculations make the assumption that the data sample is representative of the entire database.

Byte Distribution Analysis Report

The Byte Distribution Analysis Report is produced when the Byte Distribution Analysis program (SFEPBDA) is executed. This program must be executed for every extracted data sample before the Test Compression program can be executed. It must be executed again after the user modifies any RDL for a segment.

Heading Section

The heading section identifies the database, the installation, and the date and time the report was generated.

BYTE DISTRIBUTION ANALYSIS REPORT FOR IMS DBD SAMPLEDB -- MMM TEST 523				AS OF 01/26/94 AT 14:46		PAGE: 1	
STERLING SOFTWARE, INC.							
DATA SAMPLE EXTRACT INFORMATION:							
ORGANIZATION IS HDAM -- DBD(SAMPLEDB)							
DSNAME OF EXTRACT DATA SET		SOURCE OF DATA SAMPLE		DATE		TIME	
-----		-----		-----		-----	
1. SHRINK.EXTRACT.FILE		SHRINK.SAMPLE.DATABASE		01/26/94		14:45	

Figure 10-11. Heading Section

BYTE DISTRIBUTION ANALYSIS REPORT FOR IMS DBD—The title of the report, and the name of the data set.

company name—The name of the company as entered during the install process.

AS OF—The date the report was generated.

AT—The time the report was generated.

DATA SAMPLE EXTRACT INFORMATION—The type of data set from which the data was extracted, the DBD name, the name of the extract data set, the source data set, and the date and time of the extraction.

ORGANIZATION IS—The organization type of the data set.

DBD—The name of the DBD

DSNAME OF EXTRACT DATA SET—The name of the data set that contains the extracted data.

SOURCE OF DATA SAMPLE—The name of the data set that was the source of the extracted data.

DATE—The date that the data was extracted.

TIME—The time that the data was extracted.

SFS—The segment is to use the FDT of another segment because it has the Same Field Structure as the other segment. The segment which will be used to generate the FDT will have a SHARE value of NO.

YES—Two or more segments with the same segment format (fixed or variable) and the same key length are to use the same FDT. When this form of FDT sharing is used, the only RDL permitted for the compressible data is C1VER. When SOURCE OF RDL field refers to the current segment, it means that this is the segment which will be used to generate the FDT. When the SOURCE OF RDL field refers to another segment, it means that the current segment cannot be used to generate an FDT but that it will use the FDT generated for the specified segment.

PAD—The amount of free space, if any, to be left after the segment has been compressed. There are two types of padding.

N—The specified number of bytes will be added to the overall length of the compressed segment. This type of padding can be from 1 byte to 255 bytes.

P—The specified percent of the saved bytes is to be added to the overall length of the compressed segment. This means that if a pad of 10 Percent was specified and the segment was reduced by 80 bytes, 8 bytes would be added to the overall length of the compressed segment.

MIN SIZE—The minimum size of the compressed segment. Zero has the special meaning that the segment was compressed to its smallest possible size

EXTRACT RDL—The RDL shown is the RDL used to calculate the Byte Distribution.

Test Compression Report

The Test Compression Report provides the user with the information needed to determine for each segment in the database whether that segment should be compressed, and, if so, what type of compression ratio can be expected from each type of compression. This report is divided into five major sections.

1. Heading.
2. Compression Test Information.
3. DASD Utilization.
4. DASD Allocation Recommendations.
5. Management Summary.

Heading Section

The heading section identifies the database, the installation, and the date and time the report was generated.

```
COMPRESSION TEST REPORT FOR IMS DBD SAMPLEDB -- MMM TEST 523          TEST NUMBER: 1
STERLING SOFTWARE, INC.          AS OF 01/26/94 AT 14:46          PAGE: 1
DATA SAMPLE EXTRACT INFORMATION:
  ORGANIZATION IS HDAM  -- DBD(SAMPLEDB)
  DSNAME OF EXTRACT DATA SET          SOURCE OF DATA SAMPLE          DATE    TIME    SAMPLE SIZE
-----
  1. SHRINK.EXTRACT.FILE          SHRINK.SAMPLE.DATABASE          01/26/94 14:45          28
```

Figure 10-13. Heading for IMS Full Function Database—Single DSG

COMPRESSION TEST REPORT FOR IMS DBD—The title of the report, and the name of the data set.

TEST NUMBER—The number of the test.

company name—The name of the company as entered during the install process.

AS OF—The date the report was generated.

AT—The time the report was generated.

DATA SAMPLE EXTRACT INFORMATION—The type of data set from which the data was extracted, the DBD name, the name of the extract data set, the source data set, the date and time of the extraction, and the sample size.

ORGANIZATION IS—The organization type of the data set.

DBD—The name of the DBD

DSNAME OF EXTRACT DATA SET—The name of the data set that contains the extracted data.

SOURCE OF DATA SAMPLE—The name of the data set that was the source of the extracted data.

DATE—The date that the data was extracted.

TIME—The date that the data was extracted.

SAMPLE SIZE—The number of segments in the sample.

Heading Section

The heading section identifies the database, the installation, and the date and time the report was generated.

```

COMPRESSION TEST REPORT FOR IMS DBD SAMPLEDM -- MMM TEST 523          TEST NUMBER: 2
STERLING SOFTWARE, INC.                                           AS OF 01/26/94 AT 14:48          PAGE: 1
DATA SAMPLE EXTRACT INFORMATION:
  ORGANIZATION IS HDAM -- DBD(SAMPLEDM)
  DSNNAME OF EXTRACT DATA SET          SOURCE OF DATA SAMPLE          DATE    TIME    SAMPLE SIZE
-----
  1.                                     SHRINK.SAMPLE.DATABAS1
  2.                                     SHRINK.SAMPLE.DATABAS2
    
```

Figure 10-14. Heading for IMS Full Function Database—Multiple DSGs

COMPRESSION TEST REPORT FOR IMS DBD—The title of the report, and the name of the data set.

TEST NUMBER—The number of the test.

company name—The name of the company as entered during the install process.

AS OF—The date the report was generated.

AT—The time the report was generated.

DATA SAMPLE EXTRACT INFORMATION—The type of data set from which the data sample was extracted, the DBD name, the name of the extract data set, the source data set, the date and time of the extraction, and the sample size.

ORGANIZATION IS—The organization type of the data set.

DBD—The name of the DBD

DSNAME OF EXTRACT DATA SET—The name of the data set that contains the extracted data.

SOURCE OF DATA SAMPLE—The name of the data set that was the source of the extracted data.

DATE—The date that the data set was extracted.

TIME—The date that the data set was extracted.

SAMPLE SIZE—The number of segments in the sample.

Compression Test Information Section

The Compression Test Information section will be reported for each segment type that was extracted into the data sample. It indicates how effective the specified RDL and FDT control parameters were in compressing the segment.

```

* COMPRESSION TEST INFORMATION FOR SEGMENT          1 SEGMENTA *****
* NUM OF SEGS <-- SEGMENT LENGTHS -->          NUM OF BYTES          TOTAL          AVERAGE LENGTHS          <-- PERCENT OF COMPRESSION -->          BEST *
* IN SAMPLE  PREFIX  KEY  IC  DATA          IN SAMPLE          BYTES SAVED          UNCPRSD          CMPRSD          W/PREFIX          KEY+DATA          DATA          TEST *
* -----
*           3      18  10  3    90          300          100
*           HUFFMAN COMPRESSION --          168          44          47.5          56.0          62.3          1 *
*           EXPRESS COMPRESSION --          121          60          33.9          40.0          44.5          *
* FDT DATA: SOURCE OF RDL(SAMPLEDB/SEGMENTA)    PADDING=(N,000)    MINSIZE=000
*           +-----+
* EXTRACT RDL | N10,
*           | C1P90.
*           +-----+
*****
    
```

Figure 10-15. Compression Test Information Section—IMS Database

COMPRESSION TEST INFORMATION—The title of the report.

FOR SEGMENT—The segment identity.

NUM OF SEGS IN SAMPLE—The number of segments that comprised the data sample.

SEGMENT LENGTHS—Information about the length of the segment.

PREFIX—The length of the IMS segment prefix.

KEY—The number of bytes in the segment key. For an un-keyed segment, this field will be blank.

IC—This field will always be 3. It is a control field and only appears in a compressed segment.

DATA—The number of bytes of compressible data in the segment.

NUM OF BYTES IN SAMPLE—The size of the data sample expressed in bytes of data.

TOTAL BYTES SAVED—The number of bytes saved by the compression process upon which line the number appears.

AVERAGE LENGTHS—The average lengths in uncompressed and compressed form.

UNCPRSD—The average length of the uncompressed segment.

CMPRSD—The average length of the segment after being compressed by the process upon the line that the number is found.

PERCENT OF COMPRESSION—The percent of compression achieved for the the following types.

W/PREFIX—It reflects the true percent of compression that can be expected for the segment. It is based upon the total IMS length of the segment and includes the IMS prefix length as well as the compressed segment length.

KEY+DATA—It reflects the percent of compression that can be expected for the segment. It includes the non-compressible key, if any, and compressible data area.

DATA—The field reflects how well the compressible portion of the segment could be compressed. For Huffman compression, it reflects how effective the RDL was in compressing the segment. By changing the RDL, the Huffman compression percent will vary. Due to the nature of the Express compression, it will never vary for a given data sample.

BEST TEST—This field reflects the test number which gave the smallest average compressed segment length for Huffman compression. The number is incremented each run where the average length is smaller than or equal to the previous smallest average segment length.

HUFFMAN COMPRESSION—The compression that would be achieved if the segment is compressed using the RDL and FDT shown.

EXPRESS COMPRESSION—The compression that would be achieved if the segment is compressed using Express.

FDT DATA—FDT data only applies to Huffman compression.

SOURCE OF RDL—This field has meaning only if the segment uses another segment's RDL and byte distribution data for compression.

PADDING—This field indicates the type and amount of padding that has been applied to the compressed segment.

MINSIZE—This field indicates the minimum size of the compressed segment that will be written. A value of 0 indicates that the segment is to be compressed to its smallest possible size. A non-zero MINSIZE should be greater than the number of bytes from the beginning of the segment through the end of the segment key, if any, plus 5 bytes.

EXTRACT RDL—The RDL information only applies to Huffman compression.

DASD Utilization Section

The DASD Utilization section describes the current characteristics of each data set in a database. ISAM/OSAM is not supported.

Multiple DSGs Database—OSAM

```

DATASET GROUP TOTALS:
THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS          8,819 BYTES INCLUDING SEGMENT PREFIXES.
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS             5,968 BYTES INCLUDING SEGMENT PREFIXES.
1. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO31
  DDNAME: PRIMARY --- INDDO31
  RAPS PER BLK -      4  RAP BLOCKS          10  RAP BYTES - 1,020
  DEVICE ----- 3380  RECD LENGTH -- 1,690  RECDs/TRACK --- 21
                                FREE BLOCK PCT -- 0  FREE BLK FREQ - 0
  DCB IS:  RECFM -- FBS  LRECL ----- 1,690  BLKSIZE ---- 1,690
          TRACKS

  -----
  CURR ALLOCATED --      30  SEGMENTS -----          1,379
  CURR USED -----      1  UNCOMPRESSED DATA BYTES --      44,325
  COMPRESS USES ---      1  COMPRESSED DATA BYTES ----      28,211
  COMPRESS SAVES --      0  COMPRESS -- BYTES SAVED --      16,114
2. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO32
  DDNAME: PRIMARY --- INDDO32
  RAPS PER BLK -      4  RAP BLOCKS          10  RAP BYTES - 1,020
  DEVICE ----- 3380  RECD LENGTH -- 1,690  RECDs/TRACK --- 21
                                FREE BLOCK PCT -- 0  FREE BLK FREQ - 0
  DCB IS:  RECFM -- FBS  LRECL ----- 1,690  BLKSIZE ---- 1,690
          TRACKS

  -----
  CURR ALLOCATED --      30  SEGMENTS -----          8,668
  CURR USED -----     10  UNCOMPRESSED DATA BYTES --      294,712
  COMPRESS USES ---      7  COMPRESSED DATA BYTES ----      160,161
  COMPRESS SAVES --      3  COMPRESS -- BYTES SAVED --      134,551
3. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO33
  DDNAME: PRIMARY --- INDDO33
  RAPS PER BLK -      4  RAP BLOCKS          10  RAP BYTES - 1,020
  DEVICE ----- 3380  RECD LENGTH -- 1,690  RECDs/TRACK --- 21
                                FREE BLOCK PCT -- 0  FREE BLK FREQ - 0
  DCB IS:  RECFM -- FBS  LRECL ----- 1,690  BLKSIZE ---- 1,690
          TRACKS

  -----
  CURR ALLOCATED --      60  SEGMENTS -----         35,657
  CURR USED -----     38  UNCOMPRESSED DATA BYTES --     1,105,958
  COMPRESS USES ---     27  COMPRESSED DATA BYTES ----     694,819
  COMPRESS SAVES --     11  COMPRESS -- BYTES SAVED --     411,139
DATABASE TOTALS:
          TRACKS

  -----
  CURR ALLOCATED      120  SEGMENTS -----         45,704
  CURR USED -----    49  UNCOMPRESSED BYTES -----     1,444,995
  COMPRESS USES -     35  COMPRESSED DATA BYTES ----     883,191
  COMPRESS SAVES -     14  COMPRESS -- BYTES SAVED --     561,804
    
```

Figure 10-16. Multiple DSGs Database—OSAM

THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS—Average uncompressed record length for the database.

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—Average compressed record length for the database.

The physical characteristics for each data set is presented by number. This area will vary depending upon the organization of the database and whether the access method is VSAM or OSAM.

DATASET—The name of the data set.

DDNAME—The type and DDNAME of the data set.

RAPS PER BLK—The number of RAPs (Root Anchor Points) in each block.

RAP BLOCKS—The number of RAP (Root Anchor Points) blocks.

RAP BYTES—The number of RAP (Root Anchor Points) bytes.

DEVICE—The type of device on which the data set is stored.

RECD LENGTH—The length of the physical database record.

RECD S/TRACK—The number of physical records per track.

FREE BLOCK PCT—The free block percentage factor for the data set.

FREE BLK FREQ—The free block frequency factor for the data set.

DCB IS—The DCB of the data set.

RECFM—The record format of the data set.

LRECL—The logical record length of the data set.

BLKSIZE—The block size of the data set.

The current DASD allocation, current DASD utilization, DASD utilization and savings based upon current data set characteristics and allocations.

CURR ALLOCATED—The number of tracks or blocks currently allocated for the data set.

SEGMENTS—The number of segments.

CURR USED—The number of tracks or blocks currently used for the data set.

UNCOMPRESSED DATA BYTES—The number of uncompressed data bytes for the data set.

COMPRESS USES—The number of tracks or blocks used by SAMS:Compress for the data set.

COMPRESSED DATA BYTES—The number of compressed data bytes for the data set.

COMPRESS SAVES—The number of tracks or blocks saved by SAMS:Compress for the data set.

COMPRESS BYTES SAVED—The number of bytes saved by SAMS:Compress for the data set.

For a database with multiple Data Set Groups, a summary of the information in the preceding items is provided for the total database.

CURR ALLOCATED—The total number of tracks or blocks currently allocated for the database.

SEGMENTS—The total number of segments in the database.

CURR USED—The total number of tracks or blocks currently used for the database.

UNCOMPRESSED DATA BYTES—The total number of uncompressed data bytes for the database.

COMPRESS USES—The total number of tracks or blocks used by SAMS:Compress for the database.

COMPRESSED DATA BYTES—The total number of compressed data bytes for the database.

COMPRESS SAVES—The total number of tracks or blocks saved by SAMS:Compress for the database.

COMPRESS -- BYTES SAVED—The total number of bytes saved by SAMS:Compress for the database.

Single DSG HIDAM Database—VSAM

DATASET GROUP TOTALS:			
THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS		8,820 BYTES INCLUDING SEGMENT PREFIXES.	
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS		5,966 BYTES INCLUDING SEGMENT PREFIXES.	
1. DATASET: INFVSAM1.SHRINK.TESTDB.INDIV11			
DDNAME: PRIMARY --- INDIV11			
DEVICE ----- 3380	CI SIZE ----- 4,096	CIs/TRACK - 10	
CURRENT -----	FREE SPACE PCT -- 0	FREE CI FREQ -- 0	
DASD UTILIZATION IS 86 PERCENT.			
TRACKS			
CURR ALLOCATED --	60	SEGMENTS -----	45,705
CURR USED -----	44	UNCOMPRESSED DATA BYTES --	1,445,045
COMPRESS USES ---	29	COMPRESSED DATA BYTES ----	882,904
COMPRESS SAVES --	15	COMPRESS -- BYTES SAVED --	562,141

Figure 10-17. Single DSG HIDAM Database—VSAM

THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS—Average uncompressed record length for the database.

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—Average compressed record length for the database.

DATASET—The name of the data set.

DDNAME—The type and DDNAME of the data set.

DEVICE—The type of device on which the data set is stored.

CI SIZE—The size of the CI (Control Interval).

CIs/TRACK—The number of CIs (Control Interval) per track.

CURRENT DASD UTILIZATION IS—The percent of DASD that is currently used.

CURRENT FREE SPACE PCT—The percent of free space currently specified.

CURRENT FREE CI FREQ—The frequency of free CIs (Control Intervals) currently specified.

CURR ALLOCATED—The number of tracks or blocks currently allocated for the data set.

SEGMENTS—The number of segments.

CURR USED—The number of tracks or blocks currently used for the data set.

UNCOMPRESSED DATA BYTES—The number of uncompressed data bytes for the data set.

COMPRESS USES—The number of tracks or blocks used by SAMS:Compress for the data set.

COMPRESSED DATA BYTES—The number of compressed data bytes for the data set.

COMPRESS SAVES—The number of tracks or blocks saved by SAMS:Compress for the data set.

COMPRESS -- BYTES SAVED—The number of bytes saved by SAMS:Compress for the data set.

HISAM Database—VSAM

```

DATASET GROUP TOTALS:
THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS      5,043 BYTES INCLUDING SEGMENT PREFIXES.
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS        3,217 BYTES INCLUDING SEGMENT PREFIXES.
1. PRIMARY DATASET:  INFVSAM1.SHRINK.TESTDB.INDSV1P
   DDNAME:           -- INDSV1P
   DEVICE  -----  3380  CI SIZE  -----  4,096  CIs/TRACK  -   10
   MAX LRECL ----  110  RECDs/CI  -----   37  RECDs/TRK  --  370
   DASD UTILIZATION IS  86 PERCENT.
OVERFLOW DATASET:  INFVSAM1.SHRINK.TESTDB.INDSV1O
   DDNAME:  OVERFLOW -- INDSV1O
   DEVICE  -----  3380  CI SIZE  -----  4,096  CIs/TRACK  -   10
   MAX LRECL ----  110  RECDs/CI  -----   37  RECDs/TRK  --  370
   DASD UTILIZATION IS  86 PERCENT.
   TRACKS
-----
CURR ALLOCATED --    90  SEGMENTS -----                29,551
CURR USED -----    41  UNCOMPRESSED DATA BYTES --    934,307
COMPRESS USES ---    21  COMPRESSED DATA BYTES ----    574,537
COMPRESS SAVES --    20  COMPRESS -- BYTES SAVED --    359,770
    
```

Figure 10-18. HISAM Database—VSAM

THE LENGTH OF THE AVG UNCOMPRESSED DB RECORD IS—The average uncompressed record length for the database.

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—The average compressed record length for the database.

PRIMARY DATASET—The name of the primary data set.

DDNAME—The type and DDNAME of the primary data set.

DEVICE—The type of device on which the primary data set is stored.

CI SIZE—The size of the CI (Control Interval) for the primary data set.

CIs/TRACK—The number of CIs (Control Interval) per track for the primary data set.

MAX LRECL—The maximum LRECL size of the primary data set.

RECDs/CI—The number of records per CI (Control Interval) for the primary data set.

RECDs/TRK—The number of records per track for the primary data set.

DASD UTILIZATION IS—The percent of DASD used by the primary data set.

OVERFLOW DATASET—The name of the overflow data set.

DDNAME—The type and DDNAME of the overflow data set.

DEVICE—The type of device on which the overflow data set is stored.

CI SIZE—The size of the CI (Control Interval) for the overflow data set.

CI/TRACK—The number of CIs (Control Interval) per track for the overflow data set.

MAX LRECL—The maximum LRECL size of the overflow data set.

RECD/CI—The number of records per CI (Control Interval) of the overflow data set.

RECD/TRK—The number of records per track for the overflow data set.

DASD UTILIZATION IS—The percent of DASD used by the overflow data set.

CURR ALLOCATED—The number of tracks or blocks currently allocated for the data set.

SEGMENTS—The number of segments.

CURR USED—The number of tracks or blocks currently used for the data set.

UNCOMPRESSED DATA BYTES—The number of uncompressed data bytes for the data set.

COMPRESS USES—The number of tracks or blocks used by SAMS:Compress for the data set.

COMPRESSED DATA BYTES—The number of compressed data bytes for the data set.

COMPRESS SAVES—The number of tracks or blocks saved by SAMS:Compress for the data set.

COMPRESS -- BYTES SAVED—The number of bytes saved by SAMS:Compress for the data set.

DASD Allocation Recommendations Section

The DASD Allocation Recommendations section reflects changes to the DASD allocations characteristics of the database. It will also contain other recommendations for databases.

- HDAM randomization parameters.
- Free space percent and free record/CI frequency
- The amount of space to be allocated in both tracks and cylinders.

Multiple DSG HDAM Database—OSAM

```

DATA SET GROUP RECOMMENDATIONS:
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS          5,968 BYTES INCLUDING SEGMENT PREFIXES.
THESE RECOMMENDATIONS WILL MAXIMIZE THE USE OF DASD SPACE.
THESE RECOMMENDATIONS MAY BE INAPPROPRIATE FOR ON-LINE APPLICATIONS.
THESE RECOMMENDATIONS ARE BASED ON THE ASSUMPTION THAT ICF CATALOGS ARE
BEING USED. THEY ARE NOT VALID FOR VSAM USER CATALOGS.
1. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO31
DDNAME: PRIMARY --- INDDO31
RAPS PER BLK -      82  RAP BLOCKS          3  RAP BYTES -      267
DEVICE ----- 3380  RECD LENGTH -- 22,528  RECDS/TRACK ---  2
RECOMMENDED -----  FREE BLOCK PCT --  0  FREE BLK FREQ -  0
DCB IS:  RECFM -- FBS  LRECL ----- 22,528  BLKSIZE ---- 22,528
DASD UTILIZATION IS 95 PERCENT
RECOMMENDED DATA SET SIZE IS 22,528
ALLOCATION IN TRACKS --      4  IN CYLINDERS --      1
2. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO32
DDNAME: PRIMARY --- INDDO32
RAPS PER BLK -      20  RAP BLOCKS          12  RAP BYTES -    1,110
DEVICE ----- 3380  RECD LENGTH -- 22,528  RECDS/TRACK ---  2
RECOMMENDED -----  FREE BLOCK PCT --  0  FREE BLK FREQ - 10
DCB IS:  RECFM -- FBS  LRECL ----- 22,528  BLKSIZE ---- 22,528
DASD UTILIZATION IS 95 PERCENT
RECOMMENDED DATA SET SIZE IS 22,528
ALLOCATION IN TRACKS --      7  IN CYLINDERS --      1
3. DATASET: INFVSAM1.SHRINK.TESTDB.INDDO33
DDNAME: PRIMARY --- INDDO33
RAPS PER BLK -       4  RAP BLOCKS          60  RAP BYTES -    5,569
DEVICE ----- 3380  RECD LENGTH -- 22,528  RECDS/TRACK ---  2
RECOMMENDED -----  FREE BLOCK PCT --  0  FREE BLK FREQ - 10
DCB IS:  RECFM -- FBS  LRECL ----- 22,528  BLKSIZE ---- 22,528
DASD UTILIZATION IS 95 PERCENT
RECOMMENDED DATA SET SIZE IS 22,528
ALLOCATION IN TRACKS --     26  IN CYLINDERS --      2
    
```

Figure 10-19. Multiple DSG HDAM Database—OSAM

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—The average compressed record length for the database.

DATASET—The name of the data set.

DDNAME—The type and DDNAME of the data set.

RAPS PER BLK—The number of RAPS (Root Anchor Points) in each block.

RAP BLOCKS—The number of RAP (Root Anchor Points) blocks.

RAP BYTES—The number of RAP (Root Anchor Points) bytes.

DEVICE—The type of device on which the data set is stored.

RECD LENGTH—The physical record length.

RECDS/TRACK—The physical records per track.

RECOMMENDED—The recommended allocation.

FREE BLOCK PCT—The recommended free block percent.

FREE BLK FREQ—The recommended free block frequency.

DCB IS—The recommended DCB for the data set.

LRECL—The recommended logical record length.

RECFM—The recommended logical record length.

BLKSIZE—The recommended block size.

DASD UTILIZATION IS—The percent of DASD utilization.

RECOMMENDED DATA SET SIZE IS—The recommended amount of DASD space for the data set in bytes.

ALLOCATION IN TRACKS—The recommended allocation for the data set in tracks.

ALLOCATION IN CYLINDERS—The recommended allocation for the data set in cylinders.

Single DSG HIDAM Database—VSAM

```

DATA SET GROUP RECOMMENDATIONS:
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS          5,966 BYTES INCLUDING SEGMENT PREFIXES.
THESE RECOMMENDATIONS WILL MAXIMIZE THE USE OF DASD SPACE.
THESE RECOMMENDATIONS MAY BE INAPPROPRIATE FOR ON-LINE APPLICATIONS.
THESE RECOMMENDATIONS ARE BASED ON THE ASSUMPTION THAT ICF CATALOGS ARE
BEING USED.  THEY ARE NOT VALID FOR VSAM USER CATALOGS.
1. DATASET: INFVSAM1.SHRINK.TESTDB.INDIV11
   DDNAME: PRIMARY INDIV11
   DEVICE ----- 3380 CI SIZE ----- 22,528 CIs/TRACK - 2
   CI SIZE ----- 22,528 FREE SPACE PCT -- 1 FREE CI FREQ -- 10
   DASD UTILIZATION IS 95 PERCENT
   RECOMMENDED DATA SET SIZE IS 22,528
   ALLOCATION IN TRACKS -- 34 IN CYLINDERS -- 3

```

Figure 10-20. Single DSG HIDAM Database—VSAM

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—The average compressed record lengths for the database.

DATASET—The name of the data set.

DDNAME—The type and DDNAME of the data set.

DEVICE—The type of device on which the data set is stored.

CI SIZE—The size of the CI (Control Interval).

CIs/TRACK—The number of CIs (Control Interval) per track.

CI SIZE—The size of the CI (Control Interval).

FREE SPACE PCT—The percent of free space.

FREE CI FREQ—The frequency of free CIs (Control Intervals).

DASD UTILIZATION IS—The percent of DASD utilization.

RECOMMENDED DATA SET SIZE IS—The recommended amount of DASD space for the data set in bytes.

ALLOCATION IN TRACKS—The recommended allocation for the data set in tracks.

ALLOCATION IN CYLINDERS—The recommended allocation for the data set in cylinders.

HISAM Database—VSAM

Note: The HISAM database using VSAM can only have a single DSG.

```

DATA SET GROUP RECOMMENDATIONS:
THE LENGTH OF THE AVG COMPRESSED DB RECORD IS          3,217 BYTES INCLUDING SEGMENT PREFIXES.
THESE RECOMMENDATIONS WILL MAXIMIZE THE USE OF DASD SPACE.
THESE RECOMMENDATIONS MAY BE INAPPROPRIATE FOR ON-LINE APPLICATIONS.
THESE RECOMMENDATIONS ARE BASED ON THE ASSUMPTION THAT ICF CATALOGS ARE
BEING USED. THEY ARE NOT VALID FOR VSAM USER CATALOGS.
  1. PRIMARY DATASET:  INFVSAM1.SHRINK.TESTDB.INDSV10
     DDNAME:  PRIMARY  --- INDSVIP
     DEVICE  -----  3380  CI SIZE ----- 22,528  CIs/TRACK -    2
     MAX LRECL ----  3,752  RECDS/CI -----    6  RECDS/TRK  --   12
     DASD UTILIZATION IS 95 PERCENT
     ALLOCATION IN TRACKS --    19  IN CYLINDERS --    2
     OVERFLOW DATASET:
     DDNAME:  OVERFLOW -- INDSVIP
     DEVICE  -----  3380  CI SIZE ----- 22,528  CIs/TRACK -    2
     MAX LRECL ----  3,752  RECDS/CI -----    6  RECDS/TRK  --   12
     DASD UTILIZATION IS 95 PERCENT
     ALLOCATION IN TRACKS --    3  IN CYLINDERS --    1
     MODIFY THE 'DATASET' MACRO PARMS FOR THE DATA SET GROUP AS SHOWN:
     BLOCK PARAMETER -- BLKFACT1 =    6  BLKFACT2 =    6
     RECORD PARAMETER -- RECLLEN1 =  3,752  RECLLEN2 =  3,752
    
```

Figure 10-21. HISAM Database—VSAM

THE LENGTH OF THE AVG COMPRESSED DB RECORD IS—The average compressed record length for the database.

PRIMARY DATASET—The name of the primary data set.

DDNAME—The type and DDNAME of the data set.

DEVICE—The type of device on which the primary data set is stored.

CI SIZE—The size of the CI (Control Interval) for the primary data set.

CIs/TRACK—The number of CIs (Control Interval) per track for the primary data set.

MAX LRECL—The maximum LRECL of the primary data set.

RECDS/CI—The number of records per CI (Control Interval) for the primary data set.

RECDS/TRK—The number of records per track for the primary data set.

DASD UTILIZATION IS—The percent of DASD used by the primary data set.

ALLOCATION IN TRACKS—The recommended allocation for the primary data set in tracks.

ALLOCATION IN CYLINDERS—The recommended allocation for the primary data set in cylinders.

OVERFLOW DATASET—The name of the overflow data set.

DDNAME—The type and DDNAME of the overflow data set.

DEVICE—The type of device on which the overflow data set is stored.

CI SIZE—The size of the CI (Control Interval) for the overflow data set.

CI/TRACK—The number of CIs (Control Interval) per track for the overflow data set.

MAX LRECL—The maximum LRECL of the overflow data set.

RECD/CI—The number of records per CI (Control Interval) of the overflow data set.

RECD/TRK—The number of records per track for the overflow data set.

DASD UTILIZATION IS—The percent of DASD used by the overflow data set.

BLKFACT1—Primary data set blocking factor.

BLKFACT2—Overflow data set blocking factor.

RECLN1—Logical record length of the primary data set.

RECLN2—Logical record length of the overflow data set.

Management Summary Section

The Management Summary section is composed of the following three sub-sections:

- DASD Usage and Savings
- CPU Performance
- Installation Computation Factors

DASD Usage and Savings Sub-Section

	DASD TRACKS	%	DASD MEGABYTES	-- COSTS IN DOLLARS --	
				PURCHASE	MNTHLY OPER
ORIGINAL ALLOCATION:					
CURRENT ALLOCATION:	60	100	3	72	4
CURRENT USED:	10	16	0	0	0
AVAIL FOR GROWTH:	50	84	3	72	4
USED SPACE SAVINGS -- FULL COMPRESSION BASED ON CURRENT ALLOCATION					
CURRENT USED:	10	100	0	0	0
ESTIMATED USED:	10	100	0	0	0
ESTIMATED SAVINGS:	0	0	0	0	0
SUGGESTED ALLOCATION BASED UPON FULL COMPRESSION					
SUGG'D ALLOCATION:	2	100	0	0	0
COMP'D DATA NEEDS:	1	50	0	0	0
AVAIL FOR GROWTH:	1	50	0	0	0
TOTAL SAVINGS:					
CURRENT ALLOCATION:	60	100	3	72	4
SUGG'D ALLOCATION:	2	3	0	0	0
EST'D DASD SAVINGS:	58	97	3	72	4

Figure 10-22. DASD Usage and Savings Sub-section

This sub-section compares two DASD usages and the unused DASD or DASD savings. Unused DASD indicates the amount of space available for growth. The allocations, usage and unused or savings are shown in tracks of DASD, percent, megabytes of DASD, cost of procuring that amount of DASD space and the amount charged for that amount of DASD each month. These are reflected in the 4 horizontal headings, which are: DASD TRACKS, %, DASD MEGABYTES, COST IN DOLLARS -- PURCHASE MNTHLY OPER.

ORIGINAL ALLOCATION—Current DASD allocation compared to current DASD needs for the uncompressed data giving the amount of DASD available for growth.

CURRENT ALLOCATION—The amount of DASD currently allocated.

CURRENT USED—The amount of DASD currently used.

AVAIL FOR GROWTH—The amount of DASD available for growth.

USED SPACE SAVINGS -- FULL COMPRESSION BASED ON CURRENT ALLOCATION—Current DASD needs for the uncompressed data compared to

the DASD needs for the compressed data giving the amount of space that SAMS:Compress can save.

CURRENT USED—The amount of DASD currently used. Taken from CURRENT USED above.

ESTIMATED USED—The amount of DASD that would be used if full compression were implemented.

ESTIMATED SAVINGS—The amount of DASD that could be saved.

SUGGESTED ALLOCATION BASED UPON FULL COMPRESSION—Suggested DASD allocations compared to the DASD needed by SAMS:Compress compressed data, when using the SAMS:Compress recommendations, and giving the amount of DASD available for growth.

SUGG'D ALLOCATION—Suggested allocation based upon full compression.

COMP'D DATA NEEDS—The amount of DASD needed by the data at full compression.

AVAIL FOR GROWTH—The amount of DASD available for growth at full compression.

TOTAL SAVINGS—Current DASD allocation compared to the recommended DASD allocation and giving the total DASD savings that may be realized.

CURRENT ALLOCATION—The current allocation from CURRENT ALLOCATION above.

SUGG'D ALLOCATION—The suggested allocation from SUGG'D ALLOCATION above.

EST'D DASD SAVINGS—The estimated DASD savings, arrived at by subtracting SUGG'D ALLOCATION from CURRENT ALLOCATION.

CPU Performance Sub-Section

COMPRESSION PERFORMANCE (COSTS ARE IN DOLLARS):			
MEGABYTES COMPRESSED PER CPU SECOND	=		3.3
CPU SECONDS FOR COMPRESSING TOTAL DATABASE	=		0
COST FOR COMPRESSING TOTAL DATABASE	=		0.00
CPU SECONDS FOR SEQNTL PROCS OF DATABASE	=		0
COST FOR SEQNTL PROCS OF DATABASE	=		0.00
AVERAGE SEGMENT LENGTH FOR DATABASE	=	121	
SEGMENTS PROCESSED IN ONE CPU SECOND	=	28,740	
CPU COST TO PROCESS 10,000 SEGMENTS --		PRIME TIME	OFF SHIFT
		<hr/>	
BATCH:		0.12	0.01
ON-LINE:		0.28	0.03

Figure 10-23. CPU Performance Sub-Section

COSTS ARE IN—The monetary unit used by the country in which the user’s installation is located.

MEGABYTES COMPRESSED PER CPU SECOND—The number of megabytes of data that can be compressed/expanded per CPU second based upon the processor speed provided by the user’s installation.

CPU SECONDS FOR COMPRESSING TOTAL DATABASE—The amount of additional time in CPU seconds that will be required to initially compress the data.

COST FOR COMPRESSING TOTAL DATABASE—The cost in CPU charges that will be incurred to initially compress the data. This cost is based upon doing the conversion at night in the batch environment.

CPU SECONDS FOR SEQNTL PROCS OF DATABASE—The amount of additional CPU time that a run will increase if the data is processed in its totality in a sequential manner. This means reading and replacing every segment.

COST FOR SEQNTL PROCS OF DATABASE—This is the additional amount of CPU charges that a run will incur when the data is processed sequentially in its totality.

AVERAGE SEGMENT LENGTH FOR DATABASE—This is the result of dividing the total number of bytes of data, including keys, in the database by the total number of segments in the database.

SEGMENTS PROCESSED IN ONE CPU SECOND—This is derived by dividing the number of bytes of data that can be processed in one CPU second by the length of the average segment.

CPU COST TO PROCESS 10,000 SEGMENTS—This is the amount that you can expect processing cost to increase due to data compression. This reflects the cost of the expansion/compression of 10,000 of the average segments. This increased cost is provided for each of the following four environments.

BATCH/PRIME TIME—A batch job executed during primary operating hours.

BATCH/OFF SHIFT—A batch job executed during non-primary operating hours.

ON-LINE/PRIME TIME—An on-line transaction executed during primary operating hours.

ON-LINE/OFF SHIFT—An on-line transaction executed during non-primary operating hours.

Installation Computation Factors Sub-Section

This sub-section shows the CPU speed and monetary factors that were supplied by the user and used during the DASD and cost calculations.

INSTALLATION COMPUTATION FACTORS: (COSTS/CHARGES ARE IN DOLLARS)		
DFP (DATA FACILITY PRODUCT) LEVEL	=	3.3
PROCESSOR 'MIPS' RATING	=	16
DASD PURCHASE COST/MEGABYTE	=	24.00
DASD USAGE CHARGE/MGBYTE/MO	=	1.25
ONE CPU HOUR - PRIME - BATCH	=	890.00
ONE CPU HOUR - PRIME - ON-LINE	=	2,200.00
ONE CPU HOUR - NIGHT - BATCH	=	89.00
ONE CPU HOUR - NIGHT - ON-LINE	=	220.00
CHARGE PER 1,000 EXCPS PRIME	=	0.50
CHARGE PER 1,000 EXCPS NIGHT	=	0.05

Figure 10-24. Installation Computation Factors Sub-Section

COSTS/CHARGES ARE IN—The monetary unit used in the country in which the installation is located.

DFP (DATA FACILITY PRODUCT) LEVEL—Each operating system performs data storage and retrieval. The storing and retrieving of the data is the job of the Data Facility Product (DFP). Different DFP levels have different capabilities. This shows the DFP level provided by the user’s installation. The rules for this DFP level were used in the DASD space calculations.

PROCESSOR 'MIPS' RATING—This reflects the speed of the average processor for the installation. It is express in Millions of Instructions Per Second (MIPS). The MIPS rate directly affects the cost of expanding and compressing the data. The average MIPS speed should be rounded to the nearest MIPS.

DASD PURCHASE COST/MEGABYTE—The amount of money the installation spends when it acquires new DASD. This cost is expressed in terms of the money spent per megabyte of DASD storage acquired.

DASD USAGE CHARGE/MGBYTE/MO—This is the amount of money the installation charges its users per MEGABYTE of DASD per month.

ONE CPU HOUR - PRIME - BATCH—The amount of money the installation charges a user for the use of one hour of CPU time during the installation’s prime operating hours to run jobs in the batch environment.

ONE CPU HOUR - PRIME - ON-LINE—The amount of money the installation charges a user for the use of one hour of CPU time during the installation’s prime operating hours to run transactions/jobs in the on-line environment.

ONE CPU HOUR - NIGHT - BATCH—The amount of money the installation charges a user for the use of one hour of CPU time during the installation’s non-prime operating hours to run jobs in the batch environment.

ONE CPU HOUR - NIGHT - ON-LINE—The amount of money the installation charges the user for the use of one hour of CPU time during the installation's non-prime operating hours to run transactions/jobs in the on-line environment.

CHARGE PER 1,000 EXCPS PRIME—The amount of money the installation charges the user for 1,000 I/O operations during the installation's prime operating hours.

CHARGE PER 1,000 EXCPS NIGHT—The amount of money the installation charges the user for 1,000 I/O operations during the installation's non-prime operating hours.

Chapter 11. Messages and Abend Codes

Overview

This chapter lists the error and warning messages produced by SAMS:Compress and is organized by the following categories:

Abend Codes

This section lists abend codes numerically, contains explanations of the associated messages, and the appropriate corrective actions.

Messages with Message IDs

This section lists messages alphanumerically by message ID number, contains explanations of the messages, and the appropriate corrective action. All these messages have the prefix SHR.

Messages Without Message IDs

This section lists alphabetically the messages issued by the VARISAM subroutines. It also lists the SEGCC messages.

For most messages, the number of the message directly corresponds to an abend code number. For example, message SHR3687 corresponds directly to abend code 3687. A few messages do not have corresponding abend codes, just as some abend codes do not have corresponding messages.

Abend Codes

Numeric prefix conventions are used to tie the abend codes to the part of the program that issued them. Below are some common prefixes.

- 01xx abend codes are issued by the Control File initialization process.
- 09xx abend codes are issued by the Interactive Dialog initialization process.
- 36xx abend codes are issued by the test compression process.
- 37xx abend codes are issued by the byte distribution analysis process.
- 38xx abend codes are issued by the data sample extract process.

When errors described by these messages occur, the program is abnormally terminated with an abend user code, with one exception: In subroutine mode, the abend can be optionally suppressed for certain error conditions by coding a return code

parameter. Error conditions for which abends can be suppressed are identified by an “RC” value on the same line following the abend user code.

Note: For IMSPASS messages, all input parameter statements of the PARMFL data set are printed on the PRINT data set. Notation conventions for highlighting errors in these messages are:

- Syntax errors are underscored by asterisks.
- Incorrect continuation of a line is shown by an asterisk in column 72.
- Parameters that are duplicates or cannot be used together are shown by asterisks under the second parameter.

Other error messages are directed to the system output writer. Any error condition causes termination of IMSPASS execution.

Abend Code	Message
0002	INVALID FILE DESCRIPTOR TABLE
Explanation:	This abend code is associated with message SHR002I. The data set defined by a TABLxx DD statement or the PDS member indicated by an SCB is not a File Descriptor Table. The job will abend and a dump will be produced.
Action:	Correct the TABLxx DD statement and resubmit the job.
0004	{SAMS:COMPRESS or SHRINK} RECORD DEFINITIONS TOO LONG
Explanation:	This abend code is associated with message SHR004I. Record definitions exceed the FDT capacity. The job will abend and a dump will be produced.
Action:	Consolidate field definitions and resubmit the job.
0005	INVALID {SAMS:COMPRESS or SHRINK} PARM
Explanation:	This abend code is associated with message SHR005I. PARM value for File Compression Utility is invalid. The job will abend but no dump will be produced.
Action:	Correct the PARM value and resubmit the job.

Abend Code	Message
0006	CANNOT OPEN ddname CANNOT OPEN OUTFILE
Explanation:	This abend code is associated with messages SHR006I. and SHR315I respectively. No ddstatement was found for the specified ddname. Probably a spelling error. The job will abend and a dump will be produced.
Action:	Correct the JCL or program as appropriate and resubmit the job.
0008	ERROR IN RECORD DEFINITIONS
Explanation:	This abend code is associated with message SHR008I. One or more errors are detected by the File Prepass Utility in user-coded RDL specifications. A descriptive message accompanies each error detected and appears in the RDL listing. The job will abend but no dump will be produced.
Action:	Correct RDL specifications and resubmit the job.
0010 (RC=4)	“fdt” CHECK BYTE MISMATCH
Explanation:	This abend code is associated with message SHR010I. Several conditions may cause this error: <ul style="list-style-type: none"> • The FDT that is used to expand the file does not correspond to the file that is expanded. • The record area address passed to the EXPAND subroutine is not the first byte of the data portion of the record. • The compressed record is modified by the user before expansion. • The RDL Position Function is used incorrectly. The job will abend with a user code of 10 and a dump will be produced.
Action:	Correct the error and resubmit the job.
0011 (RC=4)	INVALID PUFFUP CALL
Explanation:	This abend code is associated with message SHR011I. This message is issued by the PUFFUP subroutine. It indicates that the the call to PUFFUP did not contain exactly four parameters. Control is returned to the calling program after the message is printed.
Action:	See the SAMS:Compress/2 chapter for more information about the parameters. Specify them correctly, then resubmit the job.

Abend Code	Message
0012 (RC=4)	INVALID PUFFDOWN CALL
Explanation:	This abend code is associated with message SHR012I. The message is issued by the PUFFDOWN subroutine. It indicates that the call to PUFFDOWN did not contain exactly four parameters. Control is returned to the calling program after the message is printed.
Action:	See the SAMS:Compress/2 chapter for more information about the parameters. Specify them correctly, then resubmit the job.
0013	FN AND P PARMS MUTUALLY EXCLUSIVE. FN IGNORED.
Explanation:	This abend code is associated with message SHR013I. The message is issued by the File Prepass Utility program. It indicates that an FDT in load module format is incompatible with execution of the File Prepass Utility, because it is the File Prepass Utility that creates the FDT, and FDTs are always created in sequential data set format.
Action:	If you are executing PREPASS, remove the FN parm. If you are executing SHRINK, remove the P parm. After the error has been corrected, resubmit the job.
0015 (RC=4)	“fdt” REC DEFS IMPLY WRONG LENGTH
Explanation:	This abend code is associated with message SHR015I. RDL specifications do not completely define the record, or the Position Function is used improperly. The job will abend with a user code of 15, and a dump will be produced.
Action:	Correct RDL specifications and resubmit the job.
0016	‘fdtname’ ICB MISMATCH - RECORD ICB = rrr, FDT ICB = fff
Explanation:	This abend code is associated with message SHR016I. During the EXPAND process, a record was found to have been compressed by an FDT other than the current one. The value rrr is the FDT number from the record, and fff is the current FDT’s number. The record is not expanded. A user abend code of 16 may result.
Action:	Use the FDT Services of the Interactive Dialog to determine the FDT name for the FDT numbers in question.
0020 (RC=8)	“fdt” INVALID TYPE VP OR VZ FIELD
Explanation:	This abend code is associated with message SHR020I. A field defined as type VP does not contain packed decimal data, or a field defined as VZ does not contain zoned decimal data. The job will abend with a user code of 20, and a dump will be produced.
Action:	Check the RDL specification for errors, and verify that all user records are valid.

Abend Code	Message
0025	“fdt” D FIELD > 128 BYTES
Explanation:	This abend code is associated with message SHR3625. A field whose field length descriptor is coded in the “Dc” form does not contain the delimiter character within the first 128 bytes. The job will abend with a user code of 25, and a dump will be produced.
Action:	Use a different field length descriptor and resubmit the job.
0031	{INFILE or OUTFILE} DD CARD INVALID
Explanation:	This abend code is associated with message SHR031I. Self-explanatory. The job will abend with a user code of 31, but no dump will be produced.
Action:	Resubmit the job with the correct ddstatement.
0032	{INFILE or OUTFILE} PDS MEMBERNAME NOT SPECIFIED
Explanation:	This abend code is associated with message SHR032I. The INFILE or OUTFILE data set is a PDS, but the member name is not coded on the ddstatement as required. The job will abend with a user code of 32, but no dump will be produced.
Action:	Code the member name and resubmit the job.
0033	{INFILE or OUTFILE} DSCB NOT FOUND
Explanation:	This abend code is associated with message SHR033I. UNIT and VOL JCL parameters are coded, but the data set is not contained on the specified volume. The job will abend with a user code of 33, but no dump will be produced.
Action:	Use the correct UNIT and VOL parameters and resubmit the job.
0034	{INFILE or OUTFILE} BLKSIZE NOT A MULTIPLE OF LRECL
Explanation:	This abend code is associated with message SHR034I. Self-explanatory. Applies to fixed-length records only. The job will abend with a user code of 34, but no dump will be produced.
Action:	Correct the JCL and resubmit the job.
	OUTFILE DCB INFO MISSING OR INVALID
Explanation:	This abend code is associated with message SHR316I. Self-Explanatory.
Action:	Rerun after specifying valid DCB information.

Abend Code	Message
0035	{INFILE or OUTFILE} BLKSIZE TOO SMALL
Explanation:	This abend code is associated with message SHR035I. The value of the file's BLKSIZE attribute is less than the value of the file's LRECL attribute. The job will abend with a user code of 35, but no dump will be produced.
Action:	Correct the JCL and resubmit the job.
0036	OUTFILE RECFM DIFFERS FROM RECFM BEFORE COMPRESSION
Explanation:	This abend code is associated with message SHR036I. The RECFM attribute of an expanded data set must be identical to the RECFM attribute of the original uncompressed data set. The job will abend with a user code of 36, but no dump will be produced.
Action:	Correct the OUTFILE dd statement and resubmit the job.
0037	OUTFILE LRECL LESS THAN LRECL BEFORE COMPRESSION
Explanation:	This abend code is associated with message SHR037I. The LRECL attribute of an expanded data set must be at least equal to the LRECL attribute of the original uncompressed data set. The job will abend with a user code of 37, but no dump will be produced.
Action:	Correct the OUTFILE dd statement and resubmit the job.
0038	OUTFILE KEYLEN ABSENT. NO DEFAULT TAKEN.
Explanation:	This abend code is associated with message SHR038I. The data set defined by the INFILE dd statement has physical sequential organization. The data set defined by the OUTFILE dd statement has indexed sequential organization but the KEYLEN attribute is not specified. The job will abend with a user code of 38, but no dump will be produced.
Action:	Correct the OUTFILE dd statement specifying the KEYLEN attribute and resubmit the job.
0039	OUTFILE KEY NOT WITHIN TYPE N DEFINITIONS
Explanation:	This abend code is associated with message SHR039I. The ISAM/VSAM key for the compressed data set is not exempted from compression. The job will abend with a user code of 39, but no dump will be produced.
Action:	Correct user-coded RDL specifications, exempting the key from compression, and resubmit the job.

Abend Code	Message
0040	OUTFILE RKP INVALID FOR RECFM=V..
Explanation:	This abend code is associated with message SHR040I. The value of the OUTFILE data set RKP attribute is less than 4. This is invalid for RECFM=V.., because the first four bytes contain the RDW. The job will abend with a user code of 40, but no dump will be produced.
Action:	Correct the OUTFILE dd statement, specifying a value greater than 3, and resubmit the job.
0041	OUTFILE IS OUTFILE NOT SUPPORTED FOR EXPRESS
Explanation:	This abend code is associated with message SHR041I. Only VSAM or PS files may be created from Express compressed data sets.
Action:	Correct the OUTFILE dd statement specifying one of the above data set DSORG types.
0042	OUTFILE RECFM MUST BE SPECIFIED FOR EXPRESS OUTFILE
Explanation:	This abend code is associated with message SHR042I. The RECFM parameter is required for OUTFILE DD when expanding to a PS file.
Action:	Add the RECFM parameter to the OUTFILE dd statement.
0043	OUTFILE LRECL MUST BE SPECIFIED FOR EXPRESS OUTFILE
Explanation:	This abend code is associated with message SHR043I. The LRECL parameter is required for OUTFILE DD when expanding to a PS file.
Action:	Add the LRECL parameter to the OUTFILE dd statement.
0050	INVALID PARM PASSED TO EXPAND
Explanation:	This abend code is associated with message SHR050I. PARM value for the File Expansion Utility is invalid. The job will abend with a user code of 50, but no dump will be produced.
Action:	Correct the PARM value and resubmit the job.

Abend Code	Message
0101	COMPANY CONTROL CARD NOT FIRST CARD READ
Explanation:	This abend code is associated with message SHR010I. The first control card read was not the company "C" card
Action:	Correct the order of the cards and rerun the job.
0102	AUTHORITY CARD NOT SECOND CARD READ
Explanation:	This abend code is associated with message SHR0102. The second control card read was not the authority "A" card.
Action:	Correct the order of the cards and rerun the job.
	OUTFILE ISAM/VSAM BUT UNCOMPRESSED WAS SEQUENTIAL
Explanation:	This abend code is associated with message SHR102I. Issued from File Compression and File Expansion utilities. It is expected that the DSORG file attribute remains the same for the expanded and compressed versions of the file.
Action:	Correct the error and resubmit the job.

Abend Code	Message
0103	<p>SVC 26 FAILED TO GET DATA PORTION SHAREOPTIONS, R15=xx</p> <p>SVC 26 FAILED TO GET DATA PORTION SHAREOPTIONS, R15=xx</p> <p>CONTROL FILE DOES NOT HAVE SHARE OPTIONS (4,3). WITHOUT THESE SHARE OPTIONS, THE CONTROL FILE IS DESTROYED WHEN SHARED BETWEEN MULTIPLE USERS. CURRENT DATA SHARE OPTION IS (X,X). CURRENT INDEX SHARE OPTION IS (X,X).</p>
Explanation:	<p>This abend code is associated with message SHR0103E. If SVC 26 was unable to get the share options for the data and/or index components, one or both of the first two messages precedes the third message.</p> <p>SHAREOPTIONS defaulted or was incorrectly specified on the DEFINE for the control file.</p>
Action:	<p>Recreate control file with shareoptions (4,3) at the cluster level. If SVC 26 failed, contact Customer Support. Then send all output and the dump if requested.</p> <p>OUTFILE KEYLEN NOT EQ KEYLEN BEFORE COMPRESSION</p>
Explanation:	<p>This abend code is associated with message SHR103I. Issued from File Compression and File Expansion utilities. It is expected that the KEYLEN file attribute remains the same for the expanded and compressed versions of the file.</p>
Action:	<p>Correct the error and resubmit the job.</p>

Abend Code	Message
0104	END OF CONTROL CARDS BEFORE COMPANY AND AUTHORITY CARDS WERE READ
Explanation:	This abend code is associated with message SHR0104. Neither the company nor the authority cards were found.
Action:	Supply the needed control cards and rerun the job.
	OUTFILE LRECL LARGER THAN BEFORE COMPRESSION
Explanation:	This abend code is associated with message SHR104I. Issued from the File Expansion Utility. The expanded file's LRECL is expected to be identical to the LRECL of the file before compression.
Action:	Correct the error and resubmit the job.
0105	SFEANAL OPEN ERROR, ERROR = xxx
Explanation:	This abend code is associated with message SHR0105E. The OPEN failed with error code xxx. If xxx=128, the DD card is missing or misspelled, and this message was preceded by IEC130I. If xxx=188, the data set is not VSAM.
Action:	Make sure that the SFEANAL DD card is present and describes the control file. If you are unable to resolve the problem, contact Customer Support for assistance.
	OUTFILE LRECL NOT 8 LONGER THAN INFILE LRECL
Explanation:	This abend code is associated with message SHR105I. Issued from the File Compression Utility. The compressed output file's LRECL should be at least eight bytes greater than the uncompressed file's LRECL to avoid system abends in the event that a particular compressed record is longer than the uncompressed record from which it is created.
Action:	Define the compressed output data set to include the additional 8 bytes in the record length. Then resubmit the job.

Abend Code	Message
0106	SFEANAL CLOSE ERROR, ERROR = xxx
Explanation:	This abend code is associated with message SHR0106E. The CLOSE failed with error code xxx. Error code xxx=004 indicates that the file is already closed. Other conditions probably indicate damage to the catalog or device.
Action:	For xxx=004, contact Customer Support. For other conditions, contact your systems programmer.
	OUTFILE RECFM (F..) USUALLY INVALID
Explanation:	This abend code is associated with message SHR106I. Issued from the File Compression Utility. Compressed output is usually defined as RECFM=Vxx, or RECFM=U. Unless the user specifically intends RECFM=Fxx, this message denotes an error which must be corrected.
Action:	Specify RECFM=Vxx on the OUTFILE dd statement.
0107	OUTFILE RKP NOT EQ RKP BEFORE COMPRESSION
Explanation:	This abend code is associated with message SHR107I. Issued from the File Expansion Utility. The expanded file's RKP is expected to be identical to the RKP of the file before compression.
Action:	Correct the error and resubmit the job.
0108	OUTFILE RKP PROBABLY SHOULD BE nnnn
Explanation:	This abend code is associated with message SHR108I. Issued from the File Compression Utility. According to RDL specifications, the key is placed at offset nnnn in the compressed record, but the OUTFILE data set RKP indicates a different location.
Action:	Correct the error and resubmit the job.
0109	OUTFILE RKP VALUE PREVENTS RECORD DELETION
Explanation:	This abend code is associated with message SHR109I. Issued from the File Compression Utility. The RKP value is zero for fixed-length records, or less than five for variable-length records. This RKP value prevents deletion via the DCB parameter OPTCD=L.
Action:	Correct the error and resubmit the job.

Abend Code	Message
0110	OUTFILE IS SUBSYSTEM DATA SET
Explanation:	This abend code is associated with message SHR110I. This warning message indicates that the OUTFILE data set is allocated to a JES or a SAMS:Compress for MVS data set. OUTFILE will be a compressed data set. If OUTFILE is a SAMS:Compress for MVS data set, the data will be compressed twice. If OUTFILE is a JES data set, JES will try to print/punch compressed data.
Action:	Check your JCL.
0111	ERROR ON READING DATA BASE, PCB STATUS CODE xx
Explanation:	This abend code is associated with message SHR314I. The xx indicates the PCB status code. Note: In addition to the above abend, normal SAMS:Compress File Prepass Utility abends may be issued during IMSPASS execution.
Action:	None.
0800	***** ERROR ON THE SAMS:COMPRESS CONTROL FILE
Explanation:	This abend code is associated with message SHR0800E. A LOGICAL VSAM error or a DASD PHYSICAL error has occurred while the Control File was being processed.
Action:	Obtain the I/O error report and the SYSUDUMP. If the error is a LOGICAL VSAM error, contact Customer Support for assistance. If the error is a DASD PHYSICAL error, contact your own operations area for assistance.
0801	SAMS:COMPRESS DIALOG QSAM/BPAM I/O ERROR
Explanation:	This abend code is associated with message SHR0801E. A DASD PHYSICAL error has occurred while one of the Interactive Dialog programs was processing a physical sequential data set or a partitioned data set.
Action:	Obtain the I/O error report and the SYSUDUMP and contact your own operations area for assistance.

Abend Code	Message
0900	none
Explanation:	Loading of the “ISPEXEC” interface module failed.
Action:	If you are using a release of ISPF prior to 2.0, you cannot use the Interactive Dialog. If you are using ISPF 2.0 or later, contact your systems programmer for ISPF support.
0901	none
Explanation:	Loading of the “ISPLINK” interface module failed.
Action:	Contact your systems programmer for ISPF support.
0902	none
Explanation:	Opening of the Control File failed.
Action:	Obtain the snap dump of the Control File control blocks and take it to your systems programmer for VSAM support to determine the exact nature of the problem.
0903	none
Explanation:	Closing of the Control File failed.
Action:	Obtain the snap dump of the Control File control blocks and take it to your systems programmer for VSAM support to determine the exact nature of the problem. Also see the <i>VSAM Administration Macro Instruction Reference Manual</i> or the <i>MVS/DFP Macro Instructions for Data Sets</i> for more information about VSAM return codes.
0904	none
Explanation:	Unable to allocate the snap dump data set.
Action:	Obtain the Allocation Failure report and take it to your systems programmer for help. Also see the <i>MVS/XA Systems Programming Library: System Macros and Facilities, Volume I</i> , or the <i>MVS/ESA Application Development Guide: Authorized Assembler Language Programs</i> for more information about SVC 99 return codes.
0905	none
Explanation:	During the initialization of the Interactive Dialog, the information about the user’s installation must be obtained from the Control File. The record containing the needed information was not found.
Action:	Check the startup CLIST for the Interactive Dialog and insure that it points to the Control File. If it does not, correct the CLIST and re-execute. If it does, obtain an IDCAMS dump print of the Control File and contact Customer Support for assistance.

Abend Code	Message
0906	none
Explanation:	An ISPF error occurred when the primary menu panel was being displayed.
Action:	Make sure that you are using the CLIST from the correct release of the Interactive Dialog. If the CLIST is correct, contact Customer Support for assistance.
0909	none
Explanation:	The allocation of a SYSOUT file failed. The Interactive Dialog was attempting to generate a report on SYSOUT and the dynamic allocation failed.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
0910	none
Explanation:	OPEN failed for the control file. The reason code for the abend contains the hexadecimal error code from OPEN. If the reason code is x'80', the dd card is missing or misspelled.
Action:	Make sure that you are using the CLIST from the correct release of the Interactive Dialog. If the CLIST is correct, contact Customer Support for assistance.
0912	none
Explanation:	The ddname SFELIN has had its BLKSIZE overridden in the OPEN routine, which is likely to cause a linkedit failure. This was issued by program SFEMOBJR, ordinarily while trying to generate an FDT. This can only apply if OPEN is modified by the installation to override the BLKSIZE. The overriding BLKSIZE is given in the reason code.
Action:	Exempt ddname SFELIN from any vendor package that optimizes BLKSIZE for data sets.
0920	none
Explanation:	SVC 26 was unable to get the SHAREOPTIONS information for the control file. This abend follows the panel that says that the SHAREOPTIONS were incorrect, and that they were question marks. The last four digits of the reason code contain the return codes from the data and index portion of the SVC 26 calls in the form ddi.
Action:	Contact Customer Support for assistance. Send the dump if requested.

Abend Code	Message
0999	none
Explanation:	An error occurred when the error recovery (ESTAE) environment was being established.
Action:	Contact Customer Support for assistance. Send the dump if requested.
3002	none
Explanation:	A label field within the FDT does not equal "SHRINK 3." A library containing a module with the same name as the requested FDT may be concatenated ahead of the FDT library.
Action:	Check the FDT library concatenation. Also, check the FDT library and verify that the BLKSIZE is valid.
3596	UNEXPECTED I/O RETURN CODE ON SAMS:COMPRESS INTERACTIVE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3596E. The return code from an I/O to the Control File was not one of those expected.
Action:	Obtain the error report which was routed to the user's default SYSOUT class. If it is a physical error, investigate and fix the problem locally. If it is a logical error, determine if it is something that can be corrected locally. If it cannot be corrected locally, contact Customer Support for assistance.
3597	THE "SFEBFILE" DCB FAILED TO OPEN
Explanation:	This abend code is associated with message SHR3597E. The file being analyzed must be opened to obtain needed information. The file open failed for some reason.
Action:	Check the job log for any error messages. Correct the error and resubmit the job. The JCL will be found in the Interactive Dialog GENLIB with a member name of SFETAPEA.
3598	FILE "*****" DATA SET NAMES DO NOT MATCH
Explanation:	This abend code is associated with message SHR3598E. The data set name used to analyze the data set was not the same as the data set name found in the header label of the tape file. This was probably caused by the tape being reused prematurely.
Action:	Correct the catalog. Using the Interactive Dialog, delete the old information about the file and re-analyze the correct data set.

Abend Code	Message
3599	FILE “*****” NOT IN THE INTERACTIVE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3599E. The information about file “*****” was not found in the Control File. Either the job’s JCL was created by hand and an error was made or the information about the file was deleted from the Control File before the batch analysis job ran.
Action:	The JCL for the batch tape analysis job is generated and submitted when the Dialog analyzes the data set. <ul style="list-style-type: none"> • Only analyze data sets using the Interactive Dialog. • Do not delete tape file data from the Dialog Control File until after the batch analysis has completed. • If it was decided that the file was not to be considered for compression, take no action.
3600	A SAMS:COMPRESS SIMULATION SUBROUTINE TRIED TO CALL A SHRINK SERVICE
	This abend code is associated with message SHR3600E. Any of the following unnumbered messages may appear with abend code 3600.
	SAMS:COMPRESS ABEND WAS CALLED
	SAMS:COMPRESS LINKAGE WAS CALLED
	SAMS:COMPRESS RETURN WAS CALLED
	SAMS:COMPRESS FREEMAIN WAS CALLED
	SAMS:COMPRESS WTO WAS CALLED
	A SAMS:COMPRESS EXIT, 1 THRU 9 WAS TAKEN
Explanation:	This abend should never occur.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3601	AN ATTEMPT WAS MADE TO USE THE EXPAND CODE DURING A COMPRESSION TEST
Explanation:	This abend code is associated with message SHR3601E. This abend should never occur.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Abend Code	Message
3602	INVALID FILE DESCRIPTOR TABLE
Explanation:	This abend code is associated with message SHR3602E. The data set defined by a TABLxx dd statement or the PDS member indicated by an SCB is not a File Descriptor Table. The job will abend and a dump will be produced.
Action:	Correct the TABLxx dd statement and resubmit the job.
3603	none
Explanation:	An unexpected error was detected in test compression module SFPCMPR.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3604	{SAMS:COMPRESS or SHRINK} RECORD DEFINITIONS TOO LONG
Explanation:	This abend code is associated with message SHR3604E. Record definitions exceed the FDT capacity. The job will abend and a dump will be produced.
Action:	Consolidate field definitions and resubmit the job.
3610	“EXPRESS ” CHECK BYTE MISMATCH
Explanation:	This abend code is associated with message SHR3610I. Several conditions may cause this error: <ul style="list-style-type: none"> • The FDT that is used to expand the file does not correspond to the file that is expanded. • The record area address passed to the EXPAND subroutine is not the first byte of the data portion of the record. • The compressed record is modified by the user before expansion. • The RDL Position Function is used incorrectly. The job will abend with a user code of 10 and a dump will be produced.
Action:	Correct the error and resubmit the job.

Abend Code	Message
3615	RDL FOR “*****” IMPLIES WRONG LENGTH
Explanation:	This abend code is associated with message SHR3615E. RDL specifications do not completely define the record, or the Position Function is used improperly. The job will abend with a user code of 15, and a dump will be produced.
Action:	Correct RDL specifications and resubmit the job.
3620	INVALID TYPE VP OR VZ FIELD
Explanation:	This abend code is associated with message SHR3620E. A field defined as type VP does not contain packed decimal data, or a field defined as VZ does not contain zoned decimal data. The job will abend with a user code of 20, and a dump will be produced.
Action:	Check the RDL specification for errors. Also verify that all user records are valid.
3625	“*****” D FIELD > 128 BYTES
Explanation:	This abend code is associated with message SHR3625E. A field whose field length descriptor is coded in the “Dc” form does not contain the delimiter character within the first 128 bytes. The job will abend with a user code of 25, and a dump will be produced.
Action:	Use a different field length descriptor and re-execute the job.

Abend Code	Message
3684	INVALID TESTING PROCEDURE. DATA BASE DATA SETS MUST BE DEFINED BEFORE COMPRESSION TESTING.
Explanation:	<p>This abend code is associated with message SHR3684E. The correct procedure for testing is:</p> <ul style="list-style-type: none"> • Analyze the DBD structure. • Set data sample extraction parameters. • Define the data set name that belongs to each DDNAME in the data base and each associated data base. • Generate testing JCL. • Execute testing jobs. <p>This abend has occurred because one or more of these steps was not performed in the proper sequence.</p>
Action:	<p>Activate the Interactive Dialog and enter the data set names for each of the listed ddnames. Then generate new testing JCL and proceed with the compression testing.</p>
3685	NO BYTE DISTRIBUTION DATA FOR SEGMENT ***** IN DATA BASE *****
Explanation:	<p>This abend code is associated with message SHR3685. The Byte Distribution Analysis job was not run prior to running the Test Compression job.</p>
Action:	<p>Execute the Byte Distribution Analysis job and then execute the Test Compression job.</p>
3686	ERROR OCCURRED IN "SFEMFCMP"
Explanation:	<p>This abend code is associated with message SHR3686E. A compression error occurred during the Test Compression job.</p>
Action:	<p>Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.</p>
3687	COMPRESS FOR COMPONENT "*****" FAILED
Explanation:	<p>This abend code is associated with message SHR3687E. A compression error occurred during the Test Compression job.</p>
Action:	<p>Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.</p>

Abend Code	Message
3688	COMPRESSION CODE GENERATION FAILED FOR COMPONENT -- "*****"
Explanation:	This abend code is associated with message SHR3688E. An error occurred during the generation of the compression control code for the specified segment/record.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3689	NO TEST COMPRESSION CONTROL CARD FOUND
Explanation:	This abend code is associated with message SHR3689E. There was no test compression run control card found in the JCL. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.
3690	INVALID SAMS:COMPRESS INTERFACE CODE IN THE CONTROL CARD
Explanation:	This abend code is associated with message SHR3690E. The SAMS:Compress Interface code in the control card does not represent a valid SAMS:Compress Interface. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.
3691	THIS IS NOT A TEST COMPRESSION CONTROL CARD
Explanation:	This abend code is associated with message SHR3691E. The control card read is not for the Test Compression program. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.

Abend Code	Message
3692	SAMS:COMPRESS INTERFACE CODE IN CONTROL CARD DOES NOT AGREE WITH THE EXTRACT FILE
Explanation:	This abend code is associated with message SHR3692E. The SAMS:Compress Interface code in the control card does not agree with the SAMS:Compress Interface code contained in the Data Extract file. This is probably caused by someone trying to create the test compression JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the test compression JCL using the Interactive Dialog JCL generation facility.
3693	EXTRACTS FOR DIFFERENT SAMS:COMPRESS INTERFACES
Explanation:	This abend code is associated with message SHR3693E. While the test compression program can accept a file that contains the merged data from several extract runs, all of the extract files must be from a single extract program.
Action:	Check the source of the merged extract data sets and eliminate the incorrect extract data sets. Then re-execute the merge operation and the test compression job.
3694	INVALID EXTRACT RECORD TYPE
Explanation:	This abend code is associated with message SHR3694E. This indicates that the input file was not created by one of the SAMS:Compress data extraction programs.
Action:	Check your JCL. To be safe, regenerate the test compression analysis JCL using the Interactive Dialog JCL generation facilities.
3696	LOAD OF "*****" FAILED
Explanation:	This abend code is associated with message SHR3696E. The load of the indicated module failed. This can be caused if the module cannot be found in the STEPLIB or because there was not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3697	HUFFMAN TEST COMPRESSION ENDED WITH ERRORS
Explanation:	This abend code is associated with message SHR3697E. The Huffman test compression detected an internal processing error.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Abend Code	Message
3698	UNEXPECTED VSAM RETURN CODE WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3698E. An unexpected return code was received from VSAM while processing the Control File. A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3699	“*****” NOT IN THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3699E. The indicated data structure (data base or file) could not be found in the Control File. The probable cause of this is one of the following: <ul style="list-style-type: none"> • This is old JCL for a data structure that has been deleted from the Control File. • Someone tried to create test compression JCL by converting the JCL for another data structure.
Action:	Generate the test compression JCL using the Interactive Dialog JCL generation facility.
3786	DISTRIBUTION COUNTER OVERFLOW— X“****”
Explanation:	This abend code is associated with message SHR3786E. The number of occurrences of a single hexadecimal byte exceeded the physical limit of a one word binary number. The specific hexadecimal character is shown in hex notation.
Action:	Using the Interactive Dialog, modify the data extraction parameters. Reduce the number of records or segments to be extracted and re-execute the data extraction job.
3787	BDA FOR COMPONENT -- “*****” FAILED
Explanation:	This abend code is associated with message SHR3787E. An error occurred when the byte distribution analysis code was being executed for the indicated component (segment or record).
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Abend Code	Message
3788	DISTRIBUTION CODE GENERATION FAILED FOR COMPONENT -- "*****"
Explanation:	This abend code is associated with message SHR3788E. An error occurred when the byte distribution analysis code was being generated for the indicated component (segment or record).
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3789	NO BYTE DISTRIBUTION CONTROL CARD FOUND
Explanation:	This abend code is associated with message SHR3789E. No byte distribution run control card was found in the JCL. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
3790	INVALID SAMS:COMPRESS INTERFACE CODE IN THE CONTROL CARD
Explanation:	This abend code is associated with message SHR3790E. The SAMS:Compress Interface code in the control card does not represent a valid SAMS:Compress Interface. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
3791	THIS IS NOT A BYTE DISTRIBUTION CONTROL CARD
Explanation:	This abend code is associated with message SHR3791E. The control card read is not for the byte distribution program. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.

Abend Code	Message
3792	SAMS:COMPRESS INTERFACE CODE IN CONTROL CARD DOES NOT AGREE WITH THE EXTRACT FILE
Explanation:	This abend code is associated with message SHR3792E. The SAMS:Compress Interface code in the control card does not agree with the SAMS:Compress Interface code contained in the data extract file. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
3793	EXTRACTS FOR DIFFERENT SAMS:COMPRESS INTERFACES
Explanation:	This abend code is associated with message SHR3793E. While the byte distribution program can accept a file that contains the merged data from several extract runs, all of the extract files must be from a single extract program.
Action:	Check the source of the merged extract data sets and eliminate the incorrect extract data sets. Then re-execute the merge operation and the byte distribution analysis job.
3794	INVALID EXTRACT RECORD TYPE
Explanation:	This abend code is associated with message SHR3794E. This indicates that the input file was not created by one of the SAMS:Compress data extraction programs.
Action:	Check your JCL. To be safe, regenerate the byte distribution analysis JCL using the Interactive Dialog JCL generation facilities.
3795	LOAD OF "SFEMCPLD" FAILED
Explanation:	This abend code is associated with message SHR3795E. The load of the byte distribution code compiler failed. This can be caused if the module cannot be found in the STEPLIB or because there is not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Abend Code	Message
3796	LOAD OF "SFEMRDLP" FAILED
Explanation:	This abend code is associated with message SHR3796E. The load of the RDL processor module failed. This can be caused if the module cannot be found in the STEPLIB or because there is not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3797	NO "JCL" RECORD FOR "*****" IN THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3797E. A required record was not found in the Control File for the indicated data structure (data base or file). A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3798	UNEXPECTED VSAM RETURN CODE WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3798E. An unexpected return code was received from VSAM while processing the Control File. A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
3799	"*****" NOT IN THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3799E. The indicated data structure (data base or file) could not be found in the Control File. The probable cause of this is one of the following: <ul style="list-style-type: none"> • This is old JCL for a data structure that has been deleted from the Control File. • Someone tried to create byte distribution JCL by converting the JCL for another data structure.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.

Abend Code	Message
3888	NO RECORDS WERE EXTRACTED FOR THE DATA SAMPLE
Explanation:	This abend code is associated with message SHR3888E. The “BYPASS” value in the extract parameters specified in the Interactive Dialog exceeded the total number of records in the file; or, the file in which the data sample is to be extracted may be an empty data set.
Action:	Check the sample data extract parameters and verify that the file is not empty. If the problem persists, contact Customer Support for assistance.
3889	RECEIVED AN UNEXPECTED STATUS OF “***” FROM IMS
Explanation:	This abend code is associated with message SHR3889E. The indicated IMS status code was received after a GN call. The extraction of the data sample must be accomplished with read integrity.
Action:	If the status code is “GG”, it indicates that the PCB being used has a PROCOPT of “GO”. Regenerate the extract JCL using a PCB that has a PROCOPT of only “G”. For all other status codes, contact your local data base administrator for assistance.
3890	UNKNOWN SEGMENT IN DATA BASE
Explanation:	This abend code is associated with message SHR3890E. IMS returned a segment that was not in the DBD specified at the time of DBD analysis. This can be caused by one of the following: <ul style="list-style-type: none"> • The data sets specified for the data base were for the wrong version of the data base. • The DBD analyzed was not for the desired version of the data base.
Action:	If the data sets are incorrect, correct the data set name for the data base and regenerate the extract JCL. If the wrong version of the DBD was analyzed, delete the DBD from the Control File and start the DBD analysis process from the correct DBDLIB.
3891	THE DL/I LANGUAGE INTERFACE WAS NOT FOUND
Explanation:	This abend code is associated with message SHR3891E. The DL/I language interface module DFSLI000 was not found in the STEPLIB or the LINK LIST.
Action:	If the IMS modules must be accessed using a STEPLIB, modify the skeleton SFESIMSE in the Interactive Dialog ISPSLIB data set. Add the appropriate data set to the STEPLIB dd statement, and then regenerate the JCL and re-execute the job.

Abend Code	Message
3893	PCB(**) IN PSB(*****) IS NOT FOR DBD(*****)
Explanation:	This abend code is associated with message SHR3893E. The PCB was specified to be used for extracting the sample data. This PCB was not for the data base shown.
Action:	Using the JCL generation facility of the Interactive Dialog, regenerate the extract JCL specifying a correct PCB/PSB.
3894	**** IS NOT A SUPPORTED RELEASE OF IMS
Explanation:	This abend code is associated with message SHR3894E. Extracting a data sample from an image copy of a data base can only be accomplished when using an IMS release of 1.3.0 or higher.
Action:	Regenerate the extract JCL to extract from the actual data base using standard DL/I calls.
3895	INVALID IMS RECORD TYPE ON DIALOG CONTROL FILE INVALID MVS RECORD TYPE ON DIALOG ANALYSIS FILE
Explanation:	This abend code is associated with message SHR3895E. This error indicates a logic failure in the maintenance of the Control File, or that the Control File has been damaged.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.
3896	PROGRAM "SFEPIMSE" DOES NOT WORK FOR "DEDB" DATA BASES
Explanation:	This abend code is associated with message SHR3896E. The extraction of data from a FASTPATH (DEDB) data base requires the use of program "SFEPIMSD." The probable cause of this error is that someone tried to create data extract JCL without using the JCL generation facility of the Interactive Dialog.
Action:	Generate the desired JCL using the Interactive Dialog.
3897	NO "JCL" RECORD FOR DATA BASE "*****" IN THE DIALOG CONTROL FILE NO JCL RECORD FOR FILE "*****" IN DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3897E. This indicates an error in the analysis program or the delete program, or a damaged Control File.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.

Abend Code	Message
3898	UNEXPECTED VSAM RETURN CODE WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3898E. VSAM returned an abnormal/unexecuted status code. The abend will be accompanied by a snap dump of the Control File's ACB, RPL, EXLST and I/O area.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.
3899	FILE "*****" NOT IN SAMS:COMPRESS CONTROL FILE DATABASE "*****" NOT IN DIALOG CONTROL FILE
Explanation:	This abend code is associated with message SHR3899E. The specified file or data base identifier was not found in the Control File. The probable cause of this is: <ul style="list-style-type: none"> • This is old JCL for a data base or file that has been deleted from the file. • Someone tried to create extract JCL without using the JCL generation facility of the dialog.
Action:	Generate the desired JCL using the Interactive Dialog.
3980	SEGM xxxxxxxx. FIXED SEGMENT DIDN'T EXPAND TO CORRECT LENGTH
Explanation:	This abend code is associated with message SHR350I. Express was called to expand a segment that was fixed in length. However, after expansion the segment was not the correct length.
Action:	Contact Customer Support.
3981	SEGM xxxxxxxx. NON-COMPRESSED SEGMENT LARGER THAN DEFINED
Explanation:	This abend code is associated with message SHR351I. Express was called to expand a segment. However, it was discovered that the segment wasn't compressed by Express and that the length of the segment was larger than defined in the DBD.
Action:	Contact Customer Support.

Abend Code	Message
3982	SEGM xxxxxxxx. KEY LOCATION INVALID
Explanation:	This abend code is associated with message SHR352I. Express was called to compress or expand a variable-length keyed segment. The location of the segment's key included part, or all, of the segment's length field.
Action:	Correct the key location in the DBD for that segment or, if the length is desired as part of the key, remove compression from that segment.
3985	IMSHRINK. SEGM xxxxxxxx. OPEN PROCESSING HAS FAILED.
Explanation:	This abend code is associated with message SHR308I. Self-explanatory.
Action:	Check prior messages for reason.
3986	IMSHRINK. SEGM xxxxxxxx. KEY COMPRESSION WOULD OCCUR
Explanation:	This abend code is associated with message SHR301I. Type N RDL specification does not completely contain the segment key.
Action:	Rerun IMSPASS with corrected RDL.
3987	IMSHRINK. SEGM xxxxxxxx. "SHR" REQUEST IN SEGCC BUT NO TYPE C1
Explanation:	This abend code is associated with message SHR303I. The specified segment cannot share the FDT unless type C1 is defined in RDL associated with the FDT.
Action:	Correct the error and resubmit the job.
3988	IMSHRINK. LOAD FAILED FOR SHRINKCB MODULE
Explanation:	This abend code is associated with message SHR306I. The SHRINKCB module is probably not yet on STEPLIB, JOBLIB, etc.
Action:	Correct the error and resubmit the job.
3989	IMSHRINK. INVALID SHRINKCB MODULE
Explanation:	This abend code is associated with message SHR305I. The loaded module does not have the word SHRINKCB as its first 8 characters.
Action:	Correct the error and resubmit the job.

Abend Code	Message
3990	IMSHRINK. TOO MANY FDTs
Explanation:	This abend code is associated with message SHR304I. The default is 350 FDTs.
Action:	Change the SHRMAXN EQU and reassemble, relink as non-re-entrant.
3991	DATA COMPRESSION EXIT RE-ENTERED FOR SAME SEGMENT TYPE
Explanation:	This abend code is associated with message SHR302I. A given transaction was compressing/expanding a segment while another transaction was attempting to compress/expand the same segment. This condition can only occur in a data communications environment.
Action:	Retry the transaction.
3992	none
Explanation:	The SEGM macro specified IMSHRINK as the COMPRTN, but no corresponding SEGCC macro was generated.
Action:	none

Abend Code	Message
3993	<p>This abend code is associated with the messages and message numbers shown below.</p> <p>'xxxxxxx' RECORD WAS NOT COMPRESSED BY THIS FDT</p>
Explanation:	This abend code is associated with message SHR335I. The ICB in the FDT is not the same as the ICB in the Segment. The application is using a different FDT to expand the data than the one used to initially compress the data.
Action:	Check the FDT that the application is using. Resubmit the job with the correct FDT.
	SEGM xxxxxxxx LARGER THAN EXPANSION WORK AREA
Explanation:	This abend code is also associated with message SHR331I. After segment expansion, the expanded length was greater than the maximum length specified in the DBD.
Action:	Verify that the program is accessing the correct DBD and the correct FDT for the database.
	SEGM xxxxxxxx. SEGM LENGTH IS ZERO OR NEGATIVE
Explanation:	This abend code is associated with message SHR330I. An invalid segment length was encountered. This error can occur in a data base level sharing environment, where subsystem ALPHA is updating a segment while subsystem BETA is reading the same segment.
Action:	Make sure the PSB PROCOPT of the subsystem doing the inquiry is "GOT" or "GON".
	SEGM xxxxxxxx. SHRINK OR EXPAND ERROR
	and
	xxxxxxx RECORD DEFINITIONS IMPLY WRONG LENGTH
	or
	xxxxxxx CHECK BYTE MISMATCH
Explanation:	This abend code is associated with messages SHR320I and SHR309I.
Action:	Check the correspondence between the segment type and the File Descriptor Table.

Abend Code	Message
3994	SEGM xxxxxxxx. INVALID SEGM LENGTH AND/OR KEY LENGTH SPECIFIED
Explanation:	This abend code is associated with message SHR321I. Either the SEGM specified a length greater than 32K or the sequence field extends beyond the record length.
Action:	Correct the error and resubmit the job.
3995	SEGM xxxxxxxx. CANNOT SWITCH TO/FROM CONTROL REGION
Explanation:	This abend code is associated with message SHR322I. If the LSO=X or S parameters were specified, this message will not be written.
Action:	Contact Customer Support.
3996	SEGM xxxxxxxx. IMODULE GETMAIN ERROR or xxxxxxxIMSHRINK. DYNAMIC AREA GETMAIN FAILURE
Explanation:	This abend code is associated with messages SHR323I or SHR307I respectively. Insufficient core may have been specified.
Action:	Increase the region size
3997	MODULE=XXXXXXXX RC=XXX
Explanation:	This abend code is associated with message SHR317I. Unable to load either "SHRINKCB" or a "FDT."
Action:	Make sure link edit was successful (return code 0).
	SEGM xxxxxxxx. IMODULE LOAD FAILED FOR FILE DESCRIPTOR TABLE
Explanation:	This abend code is also associated with message SHR324I. The FDT must be a load module (created by the utility FDTLOADR) on either STEPLIB, JOBLIB or LINKLIB.
Action:	Correct the error and resubmit the job.
3998	SEGM xxxxxxxx. IMODULE DELETE FAILED
Explanation:	This abend code is associated with message SHR325I. Self-explanatory.
Action:	Contact Customer Support.
3999	none
Explanation:	An unexpected error in IMS Express occurred.
Action:	Contact Customer Support.

Messages with Message IDs

Common Error Messages

When errors described by these messages occur, the program is abnormally terminated with an abend user code, with one exception: In subroutine mode, the abend can be optionally suppressed for certain error conditions by coding a return code parameter. Error conditions for which abends can be suppressed are identified by an “RC” value on the same line following the abend user code.

Message Number	Message
SHR002I	INVALID FILE DESCRIPTOR TABLE
Explanation:	This message is associated with abend code 0002. The data set defined by a TABLxx dd statement or the PDS member indicated by an SCB is not a File Descriptor Table. The job will abend and a dump will be produced.
Action:	Correct the TABLxx dd statement and resubmit the job.
SHR004I	{SAMS:COMPRESS or SHRINK} RECORD DEFINITIONS TOO LONG
Explanation:	This message is associated with abend code 0004. Record definitions exceed the FDT capacity. The job will abend and a dump will be produced.
Action:	Consolidate field definitions and resubmit the job.
SHR005I	INVALID {SAMS:COMPRESS or SHRINK} PARM
Explanation:	This message is associated with abend code 0005. PARM value for File Compression Utility is invalid. The job will abend but no dump will be produced.
Action:	Correct the PARM value and resubmit the job.
SHR006I	CANNOT OPEN ddname
Explanation:	This message is associated with abend code 0006. No dd statement was found for the specified ddname. Probably a spelling error. The job will abend and a dump will be produced.
Action:	Correct the JCL or program as appropriate and resubmit the job.
SHR008I	ERROR IN RECORD DEFINITIONS
Explanation:	This message is associated with abend code 0008. One or more errors are detected by the File Prepass Utility in user-coded RDL specifications. A descriptive message accompanies each error detected and appears in the RDL listing. The job will abend but no dump will be produced.
Action:	Correct RDL specifications and resubmit the job.

Message Number	Message
SHR010I	“fdt” CHECK BYTE MISMATCH
Explanation:	<p>This message is associated with abend code 0010 (RC=4). Several conditions may cause this error:</p> <ul style="list-style-type: none"> • The FDT that is used to expand the file does not correspond to the file that is expanded. • The record area address passed to the EXPAND subroutine is not the first byte of the data portion of the record. • The compressed record is modified by the user before expansion. • The RDL Position Function is used incorrectly. <p>The job will abend with a user code of 10 and a dump will be produced.</p>
Action:	Correct the error and resubmit the job.
SHR011I	INVALID PUFFUP CALL
Explanation:	<p>This message is associated with abend code 0011 (RC=4). The message is issued by the PUFFUP subroutine. It indicates that the call to PUFFUP did not contain exactly four parameters. Control is returned to the calling program after the message is printed.</p>
Action:	See the SAMS:Compress/2 chapter for more information about the parameters. Specify them correctly then resubmit the job.
SHR012I	INVALID PUFFDOWN CALL
Explanation:	<p>This message is associated with abend code 0012 (RC=4). The message is issued by the PUFFDOWN subroutine. It indicates that the call to PUFFDOWN did not contain exactly four parameters. Control is returned to the calling program after the message is printed.</p>
Action:	See the SAMS:Compress/2 chapter for more information about the parameters. Specify them correctly then resubmit the job.
SHR013I	FN AND P PARMS MUTUALLY EXCLUSIVE. FN IGNORED.
Explanation:	<p>This message is associated with abend code 0013. The message is issued by the File Prepass Utility program. It indicates that an FDT in load module format is incompatible with execution of the File Prepass Utility, because it is the File Prepass Utility that creates the FDT, and FDTs are always created in sequential data set format.</p>
Action:	If you are executing PREPASS, remove the FN parm. If you are executing COMPRESS, remove the P parm. After the error has been corrected, resubmit the job.

Message Number	Message
SHR015I	“fdt” REC DEFS IMPLY WRONG LENGTH
Explanation:	This message is associated with abend code 0015 (RC=4). RDL specifications do not completely define the record, or the Position Function is used improperly. The job will abend with a user code of 15, and a dump will be produced.
Action:	Correct RDL specifications and resubmit the job.
SHR016I	'fdtname' ICB MISMATCH - RECORD ICB = rrr, FDT ICB = fff
Explanation:	This message is associated with abend code 0016. During the EXPAND process, a record was discovered to have been compressed by an FDT other than the current one. The value rrr is the FDT number from the record, and fff is the current FDT's number. The record is not expanded. A user abend code of 16 may result.
Action:	Use the FDT Services of the Interactive Dialog to determine the FDT name for the FDT numbers in question.
SHR020I	“fdt” INVALID TYPE VP OR VZ FIELD
Explanation:	This message is associated with abend code 0020 (RC=8). A field defined as type VP does not contain packed decimal data, or a field defined as VZ does not contain zoned decimal data. The job will abend with a user code of 20, and a dump will be produced.
Action:	Check the RDL specification for errors. Also verify that all user records are valid.
SHR025I	“fdt” D FIELD > 128 BYTES
Explanation:	This message is associated with abend code 0025 (RC=4). A field whose field length descriptor is coded in the “Dc” form does not contain the delimiter character within the first 128 bytes. The job will abend with a user code of 25, and a dump will be produced.
Action:	Use a different field length descriptor and resubmit the job.
SHR031I	{INFILE or OUTFILE} DD CARD INVALID
Explanation:	This message is associated with abend code 0031. Self-explanatory. The job will abend with a user code of 31, but no dump will be produced.
Action:	Resubmit the job with the correct dd statement.

Message Number	Message
SHR032I	{INFILE or OUTFILE} PDS MEMBERNAME NOT SPECIFIED
Explanation:	This message is associated with abend code 0032. The INFILE or OUTFILE data set is a PDS, but the member name is not coded on the dd statement as required. The job will abend with a user code of 32, but no dump will be produced.
Action:	Code the member name and resubmit the job.
SHR033I	{INFILE or OUTFILE} DSCB NOT FOUND
Explanation:	This message is associated with abend code 0033. UNIT and VOL JCL parameters are coded, but the data set is not contained on the specified volume. The job will abend with a user code of 33, but no dump will be produced.
Action:	Provide the correct UNIT and VOL parameters and resubmit the job.
SHR034I	{INFILE or OUTFILE} BLKSIZE NOT A MULTIPLE OF LRECL
Explanation:	This message is associated with abend code 0034. Self-explanatory. Applies to fixed-length records only. The job will abend with a user code of 34, but no dump will be produced.
Action:	Correct the JCL and resubmit the job.
SHR035I	{INFILE or OUTFILE} BLKSIZE TOO SMALL
Explanation:	This message is associated with abend code 0035. The value of the file's BLKSIZE attribute is less than the value of the file's LRECL attribute. The job will abend with a user code of 35, but no dump will be produced.
Action:	Correct the JCL and resubmit the job.
SHR036I	OUTFILE RECFM DIFFERS FROM RECFM BEFORE COMPRESSION
Explanation:	This message is associated with abend code 0036. The RECFM attribute of an expanded data set must be identical to the RECFM attribute of the original uncompressed data set. The job will abend with a user code of 36, but no dump will be produced.
Action:	Correct the OUTFILE dd statement and resubmit the job.

Message Number	Message
SHR037I	OUTFILE LRECL LESS THAN LRECL BEFORE COMPRESSION
Explanation:	This message is associated with abend code 0037. The LRECL attribute of an expanded data set must be at least equal to the LRECL attribute of the original uncompressed data set. The job will abend with a user code of 37, but no dump will be produced.
Action:	Correct the OUTFILE dd statement and resubmit the job.
SHR038I	OUTFILE KEYLEN ABSENT. NO DEFAULT TAKEN.
Explanation:	This message is associated with abend code 0038. The data set defined by the INFILE dd statement has physical sequential organization. The data set defined by the OUTFILE dd statement has indexed sequential organization but the KEYLEN attribute is not specified. The job will abend with a user code of 38, but no dump will be produced.
Action:	Correct the OUTFILE dd statement specifying the KEYLEN attribute and resubmit the job.
SHR039I	OUTFILE KEY NOT WITHIN TYPE N DEFINITIONS
Explanation:	This message is associated with abend code 0039. The ISAM/VSAM key for the compressed data set is not exempted from compression. The job will abend with a user code of 39, but no dump will be produced.
Action:	Correct user-coded RDL specifications, exempting the key from compression, and resubmit the job.
SHR040I	OUTFILE RKP INVALID FOR RECFM=V..
Explanation:	This message is associated with abend code 0040. The value of the OUTFILE data set RKP attribute is less than 4. This is invalid for RECFM=V., because the first four bytes contain the RDW. The job will abend with a user code of 40, but no dump will be produced.
Action:	Correct the OUTFILE dd statement, specifying a value greater than 3, and resubmit the job.
SHR041I	OUTFILE IS OUTFILE NOT SUPPORTED FOR EXPRESS
Explanation:	This message is associated with abend code 0041. Only VSAM or PS files may be created from Express compressed data sets.
Action:	Correct the OUTFILE dd statement specifying one of the above data set DSORG types.

Message Number	Message
SHR042I	OUTFILE RECFM MUST BE SPECIFIED FOR EXPRESS OUTFILE
Explanation:	This message is associated with abend code 0042. The RECFM parameter is required for OUTFILE DD when expanding to a PS file.
Action:	Add the RECFM parameter to the OUTFILE dd statement.
SHR043I	OUTFILE LRECL MUST BE SPECIFIED FOR EXPRESS OUTFILE
Explanation:	This message is associated with abend code 0043. The LRECL parameter is required for OUTFILE DD when expanding to a PS file.
Action:	Add the LRECL parameter to the OUTFILE dd statement.
SHR050I	INVALID PARM PASSED TO EXPAND
Explanation:	This message is associated with abend code 0050. PARM value for the File Expansion Utility is invalid. The job will abend with a user code of 50, but no dump will be produced.
Action:	Correct the PARM value and resubmit the job.
SHR0101	COMPANY CONTROL CARD NOT FIRST CARD READ
Explanation:	The first control card read was not the company "C" card.
Action:	Correct the order of the cards and rerun the job.
SHR101I	INFILE KEY IS SCRAMBLED ON OUTFILE
Explanation:	Issued from the File Compression Utility. All or part of the key, as indicated by the INFILE data set's RKP and KEYLEN attributes, is compressed according to RDL specifications.
Action:	Correct the error and resubmit the job.
SHR0102	AUTHORITY CARD NOT SECOND CARD READ
Explanation:	This message is associated with abend code 0102. The second control card read was not the authority "A" card.
Action:	Correct the order of the cards and rerun the job.
SHR102I	OUTFILE ISAM/VSAM BUT UNCOMPRESSED WAS SEQUENTIAL
Explanation:	This message is associated with abend code 0102. Issued from File Compression and File Expansion utilities. It is expected that the DSORG file attribute remains the same for the expanded and compressed versions of the file.
Action:	Correct the error and resubmit the job.

Message Number	Message
SHR0103E	<p>SVC 26 FAILED TO GET DATA PORTION SHAREOPTIONS, R15=xx</p> <p>or</p> <p>SVC 26 FAILED TO GET DATA PORTION SHAREOPTIONS, R15=xx</p> <p>or</p> <p>CONTROL FILE DOES NOT HAVE SHARE OPTIONS (4,3). WITHOUT THESE SHARE OPTIONS, THE CONTROL FILE IS DESTROYED WHEN SHARED BETWEEN MULTIPLE USERS. CURRENT DATA SHARE OPTION IS (X,X). CURRENT INDEX SHARE OPTION IS (X,X).</p>
Explanation:	This message is associated with abend code 0103. If SVC 26 was unable to get the share options for the data and/or index components, one or both of the first two messages precedes the third message. The problem is that SHAREOPTIONS defaulted or was incorrectly specified on the DEFINE for the control file.
Action:	Recreate control file with shareoptions (4,3) at the cluster level. If SVC 26 failed, call Customer Support. Then mail all output and the dump if requested to do so.
SHR103I	OUTFILE KEYLEN NOT EQ KEYLEN BEFORE COMPRESSION
Explanation:	This message is associated with abend code 0103. Issued from File Compression and File Expansion utilities. It is expected that the KEYLEN file attribute remains the same for the expanded and compressed versions of the file.
Action:	Correct the error and resubmit the job.
SHR0104	END OF CONTROL CARDS BEFORE COMPANY AND AUTHORITY CARDS WERE READ
Explanation:	This message is associated with abend code 0104. Neither the company nor the authority cards were found.
Action:	Supply the needed control cards and rerun the job.
SHR104I	OUTFILE LRECL LARGER THAN BEFORE COMPRESSION
Explanation:	This message is associated with abend code 0104. Issued from the File Expansion Utility. The expanded file's LRECL is expected to be identical to the LRECL of the file before compression.
Action:	Correct the error and resubmit the job.

Message Number	Message
SHR0105E	SFEANAL OPEN ERROR, ERROR = xxx
Explanation:	This message is associated with abend code 0105. The OPEN failed with error code xxx. If xxx=128, the DD card is missing or misspelled, and this message was preceded by IEC130I. If xxx=188, the data set is not VSAM.
Action:	Make sure that the SFEANAL DD card is present and describes the control file. If you are unable to resolve the problem, contact Customer Support for assistance.
SHR105I	OUTFILE LRECL NOT 8 LONGER THAN INFILE LRECL
Explanation:	This message is associated with abend code 0105. Issued from the File Compression Utility. The compressed output file's LRECL should be at least eight bytes greater than the uncompressed file's LRECL to avoid system abends in the event that a particular compressed record is longer than the uncompressed record from which it is created.
Action:	Define the compressed output data set to include the additional 8 bytes in the record length. Then resubmit the job.
SHR0106E	SFEANAL CLOSE ERROR, ERROR = xxx
Explanation:	This message is associated with abend code 0106. The CLOSE failed with error code xxx. Error code xxx=004 indicates that the file is already closed. Other conditions probably indicate damage to the catalog or device.
Action:	For xxx=004, contact Customer Support. For other conditions, contact your systems programmer.
SHR106I	OUTFILE RECFM (F..) USUALLY INVALID
Explanation:	This message is associated with abend code 0106. Issued from the File Compression Utility. Compressed output is usually defined as RECFM=Vxx, or RECFM=U. Unless the user specifically intends RECFM=Fxx, this message denotes an error which must be corrected.
Action:	Specify RECFM=Vxx on the OUTFILE dd statement.
SHR107I	OUTFILE RKP NOT EQ RKP BEFORE COMPRESSION
Explanation:	This message is associated with abend code 0107. Issued from the File Expansion Utility. The expanded file's RKP is expected to be identical to the RKP of the file before compression.
Action:	Correct the error and resubmit the job.

Message Number	Message
SHR108I	OUTFILE RKP PROBABLY SHOULD BE nnnn
Explanation:	This message is associated with abend code 0108. Issued from the File Compression Utility. According to RDL specifications, the key is placed at offset nnnn in the compressed record, but the OUTFILE data set RKP indicates a different location.
Action:	Correct the error and resubmit the job.
SHR109I	OUTFILE RKP VALUE PREVENTS RECORD DELETION
Explanation:	This message is associated with abend code 0109. Issued from the File Compression Utility. The RKP value is zero for fixed-length records, or less than five for variable-length records. This RKP value prevents deletion via the DCB parameter OPTCD=L.
Action:	Correct the error and resubmit the job.
SHR110I	OUTFILE IS SUBSYSTEM DATA SET
Explanation:	This message is associated with abend code 0110. This warning message indicates that the OUTFILE data set is allocated to a JES or a SAMS:Compress for MVS data set. OUTFILE will be a compressed data set. If OUTFILE is a SAMS:Compress for MVS data set, the data will be compressed twice. If OUTFILE is a JES data set, JES will try to print/punch compressed data.
Action:	Check your JCL.

RDL Error Messages

RDL error messages can be issued from either the Interactive Dialog (online) or the PREPASS facility (batch) when generating the File Descriptor Table (FDT) using user-coded RDL.

The message issued by the Interactive Dialog is displayed when entering the HELP command (or pressing the <HELP> PF key) after receiving a short error message.

When the PREPASS facility abends, RDL messages appear within the listing of user-coded RDL specifications, generated by the PRINT dd statement. These messages appear in conjunction with an abend user code of 8 and the abend message, "ERROR IN RECORD DEFINITION."

In all cases, the user response is to correct the RDL specification(s) in error and execute the online or batch process once again.

SHR2011 "P" USED BUT NO PRIOR POSITION FUNCTION.

SHR202I FIXED FIELD LENGTH EXCEEDS 16K.

SHR203I FIXED FIELD LENGTH IS ZERO.

SHR204I ILLEGAL 00 IN TYPE S OR TYPE X.

SHR205I INVALID CHARACTER IN HEX STRING.

SHR206I INVALID FIELD LENGTH DESCRIPTOR.

SHR207I INVALID OPERAND IN POSITION FUNCTION.

SHR208I INVALID VARIABLE DEFINITION LENGTH.

SHR209I MISSING QUOTE AFTER REPETITION FACTOR.

SHR210I MORE THAN 15 NESTED QUOTES.

SHR211I MORE THAN 16 CONDITIONS IN GROUP.

SHR212I MORE THAN 16 VALUES IN TYPE S OR X FLD.

SHR213I NESTED PARENTHESES ILLEGAL.

SHR214I NUMBER CANNOT EXCEED 32767.

SHR215I NUMBER EXCEEDS 8 DIGITS.

SHR216I PD FIELD EXCEEDS 8 BYTES.

SHR217I RECORD DEFS EXCEED MAX SIZE.

SHR218I	RECORD DEFS WILL IMPLY WRONG LENGTH.
SHR219I	TOTAL TYPE N LENGTHS EXCEED 4095.
SHR220I	TYPE M NOT ALLOWED IN CONDITION GROUP.
SHR221I	TYPE M NOT ALLOWED IN VS“...” REP GROUP.
SHR222I	TYPE N NOT ALLOWED IN REP OR CONDIT GROUP.
SHR223I	TYPE N PRECEDED BY VARIABLE LEN FLD.
SHR224I	UNEXPECTED CHARACTER ENCOUNTERED.
SHR225I	UNPAIRED LEFT PARENTHESIS.
SHR226I	UNPAIRED LEFT QUOTE.
SHR227I	UNPAIRED QUOTE.
SHR228I	UNPAIRED QUOTE WITHIN PARENTHESES.
SHR229I	UNPAIRED RT PARENTHESIS.
SHR230I	VAR LEN ALLOWED FOR C, GA & UN FLDS ONLY.
SHR231I	VS USED WITHOUT PRIOR TYPE V DEFINITION.
SHR232I	ZERO REPETITION FACTOR.
SHR233I	“xxxxxxxx” IS AN INVALID COMPRESSION TYPE CODE.
SHR234I	A REPETITION FACTOR MUST BE A 2-DIGIT NUMBER FROM 02 THROUGH 99.
SHR235I	DATA REQUIRED TO BE NUMERIC CONTAINS NON-NUMERIC CHARACTERS.
SHR236I	S/X FIELD LENGTH MUST BE A 2-DIGIT NUMBER.
SHR237I	CONDITIONAL FIELD LENGTH MUST BE A 2-DIGIT NUMBER.
SHR238I	S/X TABLE ENTRIES MUST BE A 2-DIGIT NUMBER.
SHR239I	DATA LENGTH (xxxxxxxx) DEFINED BY RDL IS SMALLER THAN RECORD LENGTH.
SHR240I	DATA LENGTH (xxxxxxxx) DEFINED BY RDL IS LARGER THAN RECORD LENGTH.

SHR241I **LENGTH OF ENTRY VALUES FOR S OR X TABLE IS
TOO SHORT.**

SHR242I **LENGTH OF ENTRY VALUES FOR S OR X TABLE IS
TOO LONG.**

SHR243I **A CONDITION VALUE MUST HAVE THE LENGTH
SPECIFIED FOLLOWED BY A COMMA.**

SHR244I **A CONDITION VALUE MUST HAVE THE LENGTH
SPECIFIED FOLLOWED BY A COMMA.**

SAMS:Compress for IMS Error Messages

All input parameter statements of the PARMFL data set are printed on the PRINT data set. Notation for highlighting errors is:

- Syntax errors are underscored by asterisks.
- Incorrect continuation of a line is shown by an asterisk in column 72.
- Parameters that are duplicates or cannot be used together are shown by asterisks under the second parameter.

Other error messages are directed to the system output writer. Any error condition causes termination of IMSPASS execution.

Message Number	Message
SHR301I	IMSHRINK. SEGM xxxxxxxx. KEY COMPRESSION WOULD OCCUR
Explanation:	This message is associated with abend code 3986. Type N RDL specification does not completely contain the segment key.
Action:	Rerun IMSPASS with corrected RDL.
SHR302I	COMPRESSION EXIT RE-ENTERED FOR SAME SEGMENT TYPE
Explanation:	This message is associated with abend code 3991. A given transaction was compressing/expanding a segment while another transaction was attempting to compress/expand the same segment. This condition can only occur in a data communications environment.
Action:	Retry the transaction.
SHR303I	IMSHRINK. SEGM xxxxxxxx. "SHR" REQUEST IN SEGCC BUT NO TYPE C1
Explanation:	This message is associated with abend code 3987. The specified segment cannot share the FDT unless type C1 is defined in RDL associated with the FDT.
Action:	Correct the error and resubmit the job.

Message Number	Message
SHR304I	IMSHRINK. TOO MANY FDT's.
Explanation:	This message is associated with abend code 3990. The default is 350 FDTs.
Action:	Change the SHRMAXN EQU and reassemble, relink as non-re-entrant.
SHR305I	IMSHRINK. INVALID SHRINKCB MODULE
Explanation:	This message is associated with abend code 3989. The loaded module does not have the word SHRINKCB as its first 8 characters.
Action:	Correct the error and resubmit the job.
SHR306I	IMSHRINK. LOAD FAILED FOR SHRINKCB MODULE
Explanation:	This message is associated with abend code 3988. The SHRINKCB module is probably not on STEPLIB, JOBLIB, etc.
Action:	Correct the error and resubmit the job.
SHR307I	DYNAMIC AREA GETMAIN ERROR
Explanation:	This message is associated with abend code 3996. Not enough region for GETMAIN, or insufficient core specified.
Action:	Increase the region size
SHR308I	IMSHRINK. SEGM xxxxxxxx. OPEN PROCESSING HAS FAILED.
Explanation:	This message is associated with abend code 3985.
Action:	Check prior messages for reason.
SHR309I	xxxxxxx CHECK BYTE MISMATCH
Explanation:	This message is associated with abend code 3993. The message will be accompanied by message SHR320I.
Action:	Check the correspondence between the segment type and the File Descriptor Table.
SHR310I	SEGMENT xxxxxxxx.00000000 COMPRESSED. SHORTEST=00000 LONGEST=00000 AVG=00000
Explanation:	This WTO message is generated when a data base is closed. It is an informational message that can be shut off by specifying STAT=NO in the SEGCC macro in your DBD.
Action:	None.

Message Number	Message
SHR311I	SEGMENT xxxxxxxx.00000000 EXPANDED
Explanation:	This WTO message is generated when a data base is closed. It is an informational message that can be shut off by specifying STAT=NO in the SEGCC macro in your DBD.
Action:	None.
SHR312I	ANOTHER CARD EXPECTED WHEN EOF REACHED
Explanation:	Self explanatory.
Action:	None.
SHR313I	NO DDCARD FOR xxxxxxxx, PROGRAM NOT RUN
Explanation:	Self explanatory. (xxxxxxx is the required data set ddname.)
Action:	None.
SHR314I	ERROR ON READING DATA BASE, PCB STATUS CODE xx
Explanation:	This message is associated with abend code 111. The xx indicates the PCB status code. Note: In addition to the above abend, normal SAMS:Compress File Prepass Utility abends may be issued during IMSPASS execution.
Action:	None.
SHR315I	CANNOT OPEN OUTFILE
Explanation:	This message is associated with abend code 0006. Self-Explanatory.
Action:	Check SYSLOG for additional messages.
SHR316I	OUTFILE DCB INFO MISSING OR INVALID
Explanation:	This message is associated with abend code 0034. Self-Explanatory.
Action:	Rerun after specifying valid DCB information.
SHR317I	MODULE=xxxxxxx RC=xxx
Explanation:	This message is associated with abend code 3997. Unable to load either "SHRINKCB" or an "FDT."
Action:	Make sure link edit was successful (return code 0).

Message Number	Message
SHR318I	SEGM xxxxxxxx. KEY COMPRESSION REQUEST IGNORED
Explanation:	The key compression was requested in the DBD macro, SEGM.
Action:	Make sure that keys and/or sequence fields are defined as type N in the RDL.
SHR319I	SEGM xxxxxxxx. NO INIT PROCESSING SPECIFIED - DYNAMICALLY CORRECTED
Explanation:	The COMPRTN keyword parameter of the SEGM macro should read: COMPRTN=(IMSHRINK,DATA,INIT).
Action:	Correct the error and resubmit the job.
SHR320I	SEGM xxxxxxxx. SHRINK OR EXPAND ERROR
Explanation:	This message is associated with abend code 3993. This message will be accompanied by one of the following messages: xxxxxxx RECORD DEFINITIONS IMPLY WRONG LENGTH or xxxxxxx CHECK BYTE MISMATCH
Action:	Check the correspondence between the segment type and the File Descriptor Table.
SHR321I	SEGM xxxxxxxx. INVALID SEGM LENGTH AND/OR KEY LENGTH SPECIFIED
Explanation:	This message is associated with abend code 3994. Either the SEGM specified a length greater than 32K or the sequence field extends beyond the record length.
Action:	Correct the error and resubmit the job.
SHR322I	SEGM xxxxxxxx. CANNOT ISWITCH TO/FROM CONTROL REGION
Explanation:	This message is associated with abend code 3995. If the LSO=X or S parameters were specified, this message will not be written.
Action:	Contact Customer Support.

Message Number	Message
SHR323I	SEGM xxxxxxxx. IMODULE GETMAIN ERROR
Explanation:	This message is associated with abend code 3996. Not enough region for GETMAIN, or insufficient core specified.
Action:	Increase the region size
SHR324I	SEGM xxxxxxxx. IMODULE LOAD FAILED FOR FILE DESCRIPTOR TABLE
Explanation:	This message is associated with abend code 3997. The FDT must be a load module (created by the utility FDTLOADR) on either STEPLIB, JOBLIB or LINKLIB.
Action:	Correct the error and resubmit the job.
SHR325I	SEGM xxxxxxxx. IMODULE DELETE FAILED
Explanation:	This message is associated with abend code 3998.
Action:	Contact Customer Support.
SHR326I	PCB PARM MUST PRECEDE THE BEGIN PARM
Explanation:	Self explanatory.
Action:	None.
SHR327I	SEG OR P PARAMETER MISSING
Explanation:	The SEG or P parameter(s) is missing. Both parameters are required for execution.
Action:	None.
SHR328I	OUT PARM REQUIRES CORRESPONDING C OR P PARAMETER
Explanation:	OUT=C or OUT=P has no associated parameter.
Action:	None.
SHR329I	PCB NUMBER NOT IN PSB LIST OR DBDNAME NOT IN ANY PCB
Explanation:	PCB=n was coded with an invalid PCB number or PCBNAME=dbdname specified an invalid name.
Action:	None.

Message Number	Message
SHR330I	SEGM xxxxxxxx. SEGM LENGTH IS ZERO OR NEGATIVE
Explanation:	This message is associated with abend code 3993. An invalid segment length was encountered. This error can occur in a data base level sharing environment, where subsystem ALPHA is updating a segment while subsystem BETA is reading the same segment.
Action:	Make sure the PSB PROCOPT of the subsystem doing the inquiry is "GOT" or "GON".
SHR331I	SEGMENT xxxxxxxx FROM PARMCARD NOT IN PCB
Explanation:	This message is issued by the IMSPASS Program. The specified PCB does not contain this segment name.
Action:	Use PCB that is sensitive to segment.
	SEGM xxxxxxxx LARGER THAN EXPANSION WORK AREA
Explanation:	This message is associated with abend code 3993. After segment expansion, the expanded length was greater than the maximum length specified in the DBD.
Action:	Verify that the program is accessing the correct DBD and the correct FDT for the database.
SHR332I	NO DDCARD FOR OUTDB
Explanation:	Must specify //OUTDB data definition card.
Action:	Rerun job when DD CARD is specified.
SHR333I	ERROR EDITING PARM FILE, PROGRAM NOT RUN
Explanation:	Validate proper parameter specification.
Action:	Ensure valid parameters are specified.
SHR335I	'xxxxxxx' RECORD WAS NOT COMPRESSED BY THIS FDT.
Explanation:	This message is associated with abend code 3993. The ICB in the FDT is not the same as the ICB in the Segment. The application is using a different FDT to expand the data than the one used to initially compress the data.
Action:	Check the FDT that the application is using. Resubmit the job with the correct FDT.

Message Number	Message
SHR336I	SEGM xxxxxxxx. PROCOPT IS xxxx AND IS BEING UPDATED. PSB=xxxxxxx
Explanation:	This is a warning message and the job will not abend. While attempting to expand a segment, SAMS:Compress encountered an error because another application was simultaneously updating the same segment.
Action:	Due to the Processing option of GOxx, the data is returned without integrity and is not reliable. The request should be successful if you issue it again. If many of these messages are encountered, you may consider changing the Processing option to G to read with integrity.

Express Error Messages

Message Number	Message
SHR350I	SEGM xxxxxxxx. FIXED SEGMENT DIDN'T EXPAND TO CORRECT LENGTH
Explanation:	This message is associated with abend code 3980. Express was called to expand a segment that was fixed in length. However, after expansion the segment was not the correct length.
Action:	Contact Customer Support.
SHR351I	SEGM xxxxxxxx. NON-COMPRESSED SEGMENT LARGER THAN DEFINED
Explanation:	This message is associated with abend code 3981. Express was called to expand a segment. However, it was discovered that the segment wasn't compressed by Express and that the length of the segment was larger than defined in the DBD.
Action:	Contact Customer Support.
SHR352I	SEGM xxxxxxxx. KEY LOCATION INVALID
Explanation:	This message is associated with abend code 3982. Express was called to compress or expand a variable-length keyed segment. The location of the segment's key included part, or all, of the segment's length field.
Action:	Correct the key location in the DBD for that segment or, if the length is desired as part of the key, remove compression from that segment.

Interactive Dialog Error Messages and Abend Codes

Messages with abend codes numbered 01xx are issued by the Control File initialization process. Those numbered 09xx are issued by the Interactive Dialog initialization process. The test compression process issues abend codes numbered 36xx; the byte distribution analysis issues abend codes numbered 37xx; the data sample extract process issues abend codes numbered 38xx.

For more information about the Interactive Dialog, see the respective SAMS:Compress product interface chapters.

Message Number	Message
SHR800E	***** ERROR ON THE SAMS:COMPRESS DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 0800. A LOGICAL VSAM error or a DASD PHYSICAL error has occurred while the Interactive Dialog Control File was being processed.
Action:	Obtain the I/O error report and the SYSUDUMP. If the error is a LOGICAL VSAM error, contact Customer Support for assistance. If the error is a DASD PHYSICAL error, contact your own operations area for assistance.
SHR801E	SAMS:COMPRESS DIALOG QSAM/BPAM I/O ERROR
Explanation:	This message is associated with abend code 0801. A DASD PHYSICAL error has occurred while one of the Interactive Dialog programs was processing a physical sequential data set or a partitioned data set.
Action:	Obtain the I/O error report and the SYSUDUMP and contact your own operations area for assistance.
SHR3596E	UNEXPECTED I/O RETURN ON SHRINK INTERACTIVE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3596. The return from an I/O to the Interactive Dialog Control File was not one of those expected.
Action:	Obtain the error report which routed to the user's default SYSOUT class. If it is a physical error, investigate and fix the problem locally. If it is a logical error, determine if it is something that can be corrected locally. If it cannot be corrected locally, contact the SAMS:Compress Support Center for assistance.

Message Number	Message
SHR3597E	THE "SFEBFILE" DCB FAILED TO OPEN
Explanation:	This message is associated with abend code 3597. The file being analyzed must be opened to obtain needed information. The file open failed for some reason.
Action:	Check the job log for any error messages. Correct the error and resubmit the job. The JCL will be found in the Interactive Dialog GENLIB with a member name of SFETAPEA.
SHR3598E	FILE "*****" DATA SET NAMES DO NOT MATCH
Explanation:	This message is associated with abend code 3598. The data set name used to analyze the data set was not the same as the data set name found in the header label of the tape file. This was probably caused by the tape being reused prematurely.
Action:	Correct the catalog. Using the Interactive Dialog, delete the old information about the file and re-analyze the correct data set.
SHR3599E	FILE "*****" NOT IN THE SAMS:COMPRESS INTERACTIVE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3599. The information about file "*****" was not found in the Interactive Dialog Control File. Either the job's JCL was created by hand and an error was made or the information about the file was deleted from the Interactive Dialog Control File before the batch analysis job ran.
Action:	The JCL for the batch tape analysis job is generated and submitted when the Dialog analyzes the data set. <ul style="list-style-type: none"> • Only analyze data sets using the Interactive Dialog. • Do not delete tape file data from the Dialog Control File until after the batch analysis has completed. • If it was decided that the file was not to be considered for compression, take no action.
SHR3600E	A SAMS:COMPRESS SIMULATION SUBROUTINE TRIED TO CALL A SAMS:COMPRESS SERVICE
Explanation:	This message is associated with abend code 3600. This abend should never occur.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Message Number	Message
SHR3601E	AN ATTEMPT WAS MADE TO USE THE EXPAND CODE DURING A COMPRESSION TEST
Explanation:	This message is associated with abend code 3601. This abend should never occur.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3615E	RDL FOR “*****” IMPLIES WRONG LENGTH
Explanation:	This message is associated with abend code 3615. The RDL specification does not properly match the record passed.
Action:	Correct the RDL definition for the segment/record. Then re-execute the byte distribution job and the test compression job.
SHR3620E	INVALID BYTE VP OR VZ FIELD
Explanation:	This message is associated with abend code 3620. A field defined as type VP does not contain packed decimal data, or a field defined as VZ does not contain zoned decimal data.
Action:	Correct the RDL definition for the segment/record. Then re-execute the byte distribution job and the test compression job.
SHR3625E	“*****” D FIELD > 128 BYTES
Explanation:	This message is associated with abend code 3625. A field whose field length descriptor is coded in the “Dc” form does not contain the delimiter character within the within the first 128 bytes.
Action:	Use a different field length descriptor and re-execute the job.

Message Number	Message
SHR3684E	INVALID TESTING PROCEDURE. DATA BASE DATA SETS MUST BE DEFINED BEFORE COMPRESSION TESTING.
Explanation:	<p>This message is associated with abend code 3684. The correct procedure for testing is:</p> <ul style="list-style-type: none"> • Analyze the DBD structure. • Set data sample extraction parameters. • Define the data set name that belongs to each ddname in the data base and each associated data base. • Generate testing JCL. • Execute testing jobs. <p>This abend has occurred because one or more of these steps was not performed in the proper sequence.</p>
Action:	<p>Activate the Dialog and enter the data set names for each of the listed ddnames. Then generate new testing JCL and proceed with the compression testing.</p>
SHR3685E	NO BYTE DISTRIBUTION DATA FOR SEGMENT ***** IN DATA BASE *****
Explanation:	<p>This message is associated with abend code 3685. The Byte Distribution Analysis job was not run prior to running the Test Compression job.</p>
Action:	<p>Execute the Byte Distribution Analysis job and then execute the Test Compression job.</p>
SHR3686E	ERROR OCCURRED IN “SFEMFCMP”
Explanation:	<p>This message is associated with abend code 3686. A compression error occurred during the Test Compression job.</p>
Action:	<p>Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.</p>
SHR3687E	COMPRESS FOR RECORD FAILED
Explanation:	<p>This message is associated with abend code 3687. A compression error occurred during the Test Compression job.</p>
Action:	<p>Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.</p>

Message Number	Message
SHR3688E	COMPRESSION CODE GENERATION FAILED FOR RECORD
Explanation:	This message is associated with abend code 3688. An error occurred during the generation of the compression control code for the specified segment/record.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3689E	NO TEST COMPRESSION CONTROL CARD FOUND
Explanation:	This message is associated with abend code 3689. There was no test compression run control card found in the JCL. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.
SHR3690E	INVALID SAMS:COMPRESS INTERFACE CODE IN THE CONTROL CARD
Explanation:	This message is associated with abend code 3690. The Interface code in the control card does not represent a valid Interface. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.
SHR3691E	THIS IS NOT A TEST COMPRESSION CONTROL CARD
Explanation:	This message is associated with abend code 3691. The control card read is not for the Test Compression program. This was probably caused by someone trying to create the test compression JCL rather than by generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the Test Compression JCL using the Interactive Dialog JCL generation facility.

Message Number	Message
SHR3692E	SAMS:COMPRESS INTERFACE CODE IN CONTROL CARD DOES NOT AGREE WITH THE EXTRACT FILE
Explanation:	This message is associated with abend code 3692. The Interface code in the control card does not agree with the Interface code contained in the Data Extract file. This is probably caused by someone trying to create the test compression JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the test compression JCL using the Interactive Dialog JCL generation facility.
SHR3693E	EXTRACTS FOR DIFFERENT SHRINK INTERFACES
Explanation:	This message is associated with abend code 3693. While the test compression program can accept a file that contains the merged data from several extract runs, all of the extract files must be from a single extract program.
Action:	Check the source of the merged extract data sets and eliminate the incorrect extract data sets. Then re-execute the merge operation and the test compression job.
SHR3694E	INVALID EXTRACT RECORD TYPE
Explanation:	This message is associated with abend code 3694. This indicates that the input file was not created by one of the SAMS:Compress data extraction programs.
Action:	Check your JCL. To be safe, regenerate the test compression analysis JCL using the Interactive Dialog JCL generation facilities.
SHR3696E	LOAD OF "*****" FAILED
Explanation:	This message is associated with abend code 3696. The load of the indicated module failed. This can be caused if the module cannot be found in the STEPLIB or because there was not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Message Number	Message
SHR3698E	UNEXPECTED VSAM RETURN CODE WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3698. An unexpected return code was received from VSAM while processing the Interactive Dialog Control File. A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3699E	“*****” NOT IN THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3699. The indicated data structure (data base or file) could not be found in the Interactive Dialog Control File. The probable cause of this is one of the following: <ul style="list-style-type: none"> • This is old JCL for a data structure that has been deleted from the Interactive Dialog Control File. • Someone tried to create test compression JCL by converting the JCL for another data structure.
Action:	Generate the test compression JCL using the Interactive Dialog JCL generation facility.
SHR3786E	DISTRIBUTION COUNTER OVERFLOW -- X“***”
Explanation:	This message is associated with abend code 3786. The number of occurrences of a single hexadecimal byte exceeded the physical limit of a one word binary number. The specific hexadecimal character is shown in hex notation.
Action:	Using the Interactive Dialog, modify the data extraction parameters. Reduce the number of records or segments to be extracted and re-execute the data extraction job.
SHR3787E	BDA FAILED
Explanation:	This message is associated with abend code 3787. An error occurred when the byte distribution analysis code was being executed for the indicated component (segment or record).
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3788E	DISTRIBUTION CODE GENERATION FAILED
Explanation:	This message is associated with abend code 3788. An error occurred when the byte distribution analysis code was being generated for the indicated component (segment or record).
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Message Number	Message
SHR3789E	NO BYTE DISTRIBUTION CONTROL CARD FOUND
Explanation:	This message is associated with abend code 3789. No byte distribution run control card was found in the JCL. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
SHR3790E	INVALID SHRINK INTERFACE CODE IN THE CONTROL CARD
Explanation:	This message is associated with abend code 3790. The Interface code in the control card does not represent a valid Interface. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
SHR3791E	THIS IS NOT A BYTE DISTRIBUTION CONTROL CARD
Explanation:	This message is associated with abend code 3791. The control card read is not for the byte distribution program. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
SHR3792E	SHRINK INTERFACE CODE IN CONTROL CARD DOES NOT AGREE WITH THE EXTRACT FILE
Explanation:	This message is associated with abend code 3792. The Interface code in the control card does not agree with the Interface code contained in the data extract file. This is probably caused by someone trying to create the byte distribution JCL rather than generating it using the Interactive Dialog JCL generation facility.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.

Message Number	Message
SHR3793E	EXTRACTS FOR DIFFERENT SHRINK INTERFACES
Explanation:	This message is associated with abend code 3793. While the byte distribution program can accept a file that contains the merged data from several extract runs, all of the extract files must be from a single extract program.
Action:	Check the source of the merged extract data sets and eliminate the incorrect extract data sets. Then re-execute the merge operation and the byte distribution job.
SHR3794E	INVALID EXTRACT RECORD TYPE
Explanation:	This message is associated with abend code 3794. This indicates that the input file was not created by one of the SAMS:Compress data extraction programs.
Action:	Check your JCL. To be safe, regenerate the byte distribution analysis JCL using the Interactive Dialog JCL generation facilities.
SHR3795E	LOAD OF "SFEMCPLD" FAILED
Explanation:	This message is associated with abend code 3795. The load of the byte distribution code compiler failed. This can be caused if the module cannot be found in the STEPLIB or because there is not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3796E	LOAD OF "SFEMRDLP" FAILED
Explanation:	This message is associated with abend code 3796. The load of the RDL processor module failed. This can be caused if the module cannot be found in the STEPLIB or because there is not enough storage in the region to load the module.
Action:	If the load failed due to insufficient storage, increase the region size and rerun the job. If the module was not found, contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.

Message Number	Message
SHR3797E	NO JCL RECORD FOR "*****" IN THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3797. A required record was not found in the Interactive Dialog Control File for the indicated data structure (data base or file). A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3798E	UNEXPECTED VSAM RETURN WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3798. An unexpected return code was received from VSAM while processing the Interactive Dialog Control File. A snap dump of the control areas for the file will be produced.
Action:	Contact Customer Support for assistance. Then mail any I/O report and the snap dump if requested.
SHR3799E	"*****" NOT IN THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3799. The indicated data structure (data base or file) could not be found in the Interactive Dialog Control File. The probable cause of this is one of the following: <ul style="list-style-type: none"> • This is old JCL for a data structure that has been deleted from the Interactive Dialog Control File. • Someone tried to create byte distribution JCL by converting the JCL for another data structure.
Action:	Generate the byte distribution JCL using the Interactive Dialog JCL generation facility.
SHR3888E	NO RECORDS WERE EXTRACTED FOR THE DATA SAMPLE
Explanation:	This message is associated with abend code 3888. The "BYPASS" value in the extract parameters specified in the Interactive Dialog exceeded the total number of records in the file; or, the file in which the data sample is to be extracted may be an empty data set.
Action:	Check the sample data extract parameters and verify that the file is not empty. If the problem persists, contact Customer Support for assistance.

Message Number	Message
SHR3889E	RECEIVED AN UNEXPECTED STATUS OF “***” FROM IMS
Explanation:	This message is associated with abend code 3889. The indicated IMS status code was received after a GN call. The extraction of the data sample must be accomplished with read integrity.
Action:	If the status code is “GG”, it indicates that the PCB being used has a PROCOPT of “GO”. Regenerate the extract JCL using a PCB that has a PROCOPT of only “G”. For all other status codes, contact your local data base administrator for assistance.
SHR3890E	UNKNOWN SEGMENT IN DATA BASE
Explanation:	This message is associated with abend code 3890. IMS returned a segment that was not in the DBD specified at the time of DBD analysis. This can be caused by one of the following: <ul style="list-style-type: none"> • The data sets specified for the data base were for the wrong version of the data base. • The DBD analyzed was not for the desired version of the data base.
Action:	If the data sets are incorrect, correct the data set name for the data base and regenerate the extract JCL. If the wrong version of the DBD was analyzed, delete the DBD from the Interactive Dialog Control File and start the DBD analysis process from the correct DBDLIB.
SHR3891E	THE DL/I LANGUAGE INTERFACE WAS NOT FOUND
Explanation:	This message is associated with abend code 3891. The DL/I language interface module DFSLI000 was not found in the STEPLIB or the LINKLIST.
Action:	If the IMS modules must be accessed using a STEPLIB, modify the skeleton SFESIMSE in the SHRINK Interactive Dialog ISPSLIB data set. Add the appropriate data set to the STEPLIB dd statement, and then regenerate the JCL and re-execute the job.
SHR3893E	PCB(***) IN PSB(*****) IS NOT FOR DBD(*****)
Explanation:	This message is associated with abend code 3893. The PCB was specified to be used for extracting the sample data. This PCB was not for the data base shown.
Action:	Using the JCL generation facility of the Interactive Dialog, regenerate the extract JCL specifying a correct PCB/PSB.

Message Number	Message
SHR3894E	“****” IS NOT A SUPPORTED RELEASE OF IMS
Explanation:	This message is associated with abend code 3894. Extracting a data sample from an image copy of a data base can only be accomplished when using an IMS release of 1.3.0 or higher.
Action:	Regenerate the extract JCL to extract from the actual data base using standard DL/I calls.
SHR3895E	INVALID IMS RECORD TYPE ON DIALOG CONTROL FILE INVALID MVS RECORD TYPE ON DIALOG ANALYSIS FILE
Explanation:	This message is associated with abend code 3895. This error indicates a logic failure in the maintenance of the Interactive Dialog Control File, or that the Control File has been damaged.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.
SHR3896E	PROGRAM “SFEPIMSE” DOES NOT WORK FOR “DEDB” DATA BASES
Explanation:	This message is associated with abend code 3896. The extraction of data from a FAST PATH (DEDB) data base requires the use of program “SFEPIMSD”. The probable cause of this error is that someone tried to create data extract JCL without using the JCL generation facility of the Interactive Dialog.
Action:	Generate the desired JCL using the Interactive Dialog.
SHR3897E	NO “JCL” RECORD FOR DATA BASE “*****” IN THE DIALOG CONTROL FILE NO JCL RECORD FOR FILE “*****” IN DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3897. This indicates an error in the analysis program or the delete program, or a damaged Interactive Dialog CONTROL File.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.

Message Number	Message
SHR3898E	UNEXPECTED VSAM RETURN CODE WHILE ACCESSING THE DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3898. VSAM returned an abnormal/unexecuted status code. The abend will be accompanied by a snap dump of the Interactive Dialog Control File's ACB, RPL, EXLST and I/O area.
Action:	Contact Customer Support for assistance. Then mail any I/O error report and the snap dump if requested.
SHR3899E	FILE "*****" NOT IN SHRINK CONTROL FILE DATABASE "*****" NOT IN DIALOG CONTROL FILE
Explanation:	This message is associated with abend code 3899. The specified file or data base identifier was not found in the Interactive Dialog Control File. The probable cause of this is: <ul style="list-style-type: none">• This is old JCL for a data base or file that has been deleted from the file.• Someone tried to create extract JCL without using the JCL generation facility of the dialog.
Action:	Generate the desired JCL using the Interactive Dialog.

Messages Without Message IDs

This section contains the messages issued by the VARISAM subroutines and the SEGCC error messages.

VARISAM Error Messages

Each VARISAM error message is followed by the name of the subroutine issuing the message. VARISAM error messages are listed in ascending alphabetical order. Probable causes for each error are described with each message.

VARISAM does not cause any abends under error conditions. Instead, an error message is printed, and control is returned to the calling program with a return code value set to indicate which error occurred. The calling program must contain logic to handle the error condition.

Message	Subroutine
COULD NOT OPEN FILE RC=8	VISOPEN
Probable Causes: Missing DD card. The name specified by the fourth parameter did not match DDNAME.	
FILE NOT OPEN FOR I-O RC=16	VISDELETE
Probable Causes: To enable the use of this function, you must specify "I-O" to VISOPEN for this file.	
FILE OPEN FOR INPUT RC=24	VISREWRT
Probable Causes: The VISOPEN CALL for this file must specify "I-O" for the rewrite function to be valid.	
FILE OPEN FOR INPUT RC=24	VISWRITE
Probable Causes: The VISWRITE function is invalid for files opened as "INPUT". You must specify "OUTPUT" or "I-O" in the VISOPEN CALL.	
FILE OPEN FOR OUTPUT RC=24	VISREAD
Probable Causes: The VISREAD function is invalid for files opened as OUTPUT. You must specify "INPUT" or "I-O" in the VISOPEN CALL.	
FILE OPEN FOR READ OR REWRITE RC=24	VISWRITE
Probable Causes: The VISOPEN CALL specified sequential (S) I/O. You must specify it as random (R) I/O or use the VISREWRT CALL.	
FILE OPEN FOR VISWRITE ONLY RC=24	VISREWRT
Probable Causes: The VISOPEN CALL for this file must specify "I-O" for the rewrite function to be valid.	

Message	Subroutine
INVALID PARM LIST RC=4	VISDELET
Probable Causes:	Fewer than three parameters passed.
INVALID PARM LIST RC=4	VISOPEN
Probable Causes:	Fewer than five parameters passed. The second parameter is neither "INPUT", "OUTPUT", nor "I-O". The third parameter is neither "R" nor "S".
INVALID PARM LIST RC=4	VISREAD
Probable Causes:	Fewer than three parameters passed.
INVALID PARM LIST RC=4	VISREWRT
Probable Causes:	Fewer than three parameters passed.
INVALID PARM LIST RC=4	VISWRITE
Probable Causes:	Fewer than three parameters passed to VISWRITE. The KEY parameter is invalid. For random writes (R), the embedded key in the record is used.
I/O CALLED AFTER BAD VISOPEN RC=8	VISOPEN
Probable Causes:	A non-zero status was returned by VISOPEN. No further processing can be done on this file
I/O ERROR RC=12	VISDELET
Probable Causes:	An I/O error other than "record not found" (e.g., parity check) occurred.
I/O ERROR RC=12	VISREWRT
Probable Causes:	An I/O error other than those enumerated (e.g., parity check) occurred.
KEY CHANGE RC=20	VISREWRT
Probable Causes:	The key for the rewrite operation differs from the key of the record last read from the same file using a call to VISREAD.
LAST CALL NOT GOOD READ RC=24	VISREWRT
Probable Causes:	The immediately previous VARISAM call for this file was not a call to VISREAD which completed successfully.
PRESET STATUS INVALID RC=4	VISREAD
Probable Causes:	For a keyed sequential read, the STAT parameter must be preset to binary 0, 1 or 2 to control the SETL function.

Message	Subroutine
RANDOM READ REQUIRES KEY RC=4	VISREAD
Probable Causes:	The file was opened as random (R), but no key was supplied in the call to VISREAD.
RCD FORMAT NOT VARIABLE RC=12	VISOPEN
Probable Causes:	DCB specifies either fixed or undefined records. User has failed to change JCL to accommodate compressed data set.
RCD LENGTH CHANGE RC=20	VISREWRT
Probable Causes:	The file is being processed sequentially (IOTYPE coded as "S" on VISOPEN call for this file), and the length of the record presented for rewrite differs from the length of the record last successfully read. Record length changes are not permitted for update-in-place.
RECORD LENGTH LONGER THAN LRECL RC=12	VISWRITE
Probable Causes:	An attempt was made to write a record with a length greater than the length specified in the LRECL parameter of the DCB.
RKP LESS THAN 5 RC=16	VISDELET
Probable Causes:	For variable-length records, the delete byte appears immediately after the RDW, in relative position 4, and cannot overlay the key.
SHRINK ERROR RC=16	VISREAD
Probable Causes:	This message is preceded by an explanatory SAMS:Compress message.
SHRINK ERROR RC=16	VISREWRT
Probable Causes:	The message is preceded by an explanatory SAMS:Compress message.
SHRINK ERROR RC=16	VISWRITE
Probable Causes:	This message is preceded by an explanatory SAMS:Compress message.

SEGCC Error Messages (DBDGEN)

Severity	Messages/Comments
none	CONTINUATION OF segname CSECT This message will occur normally.
none	THE SEGMENT NAME ASSUMED TO BE LOADABLE FDT MEMBERNAME This is an informational message and always occurs when the FN parameter is omitted.
8	INVALID FN PARAMETER The value was not a validly constructed 8-character symbol.
8	MACRO LABEL MISSING. NEEDED TO CONTINUE CSECT The SEGCC macro needs this label to continue the control section generated by the associated SEGM macro.
8	EXTRA PARAMETER FOUND. VALUE IS "x...x" A parameter was found that is not valid. Remove it and reassem- ble.

Appendix A. Customer Support Services

Sterling Software, Inc. takes pride in creating leading edge products using state of the art software technology. Of equal importance is providing high quality technical support. The methods shown below helps us give you better, faster service. If you have any comments about software or service, please contact Customer Support. We always welcome suggestions to provide better service.

Before you Call

You can help us serve to you better by performing the following simple steps before you call Customer Support:

- Read the documentation carefully.
- Reconstruct the events and write them down.

When you Call

If you need technical programming or analysis support, call Customer Support at the following telephone.

Telephone (800) 889-0226

Support Representatives

Unless you are already working with someone on your particular problem, do not ask for a specific support representative .

Incomplete information

Incomplete information may delay or prevent a solution to your problem. If you do not respond to a request for more information within 4 weeks, we consider the problem closed. If you cannot provide the information within 4 weeks, just let us know. We'll keep the problem open.

Fix Confirmation

Confirm the results of the fix to the support representative once you receive and test it. Let us know if it solved the problem or not. If you do not respond within 2 weeks, we consider the problem closed.

Dumps

To troubleshoot problems, a full SYSMDUMP or an SVC dump is usually needed because SAMS:Compress is operating system software. An ABEND-AID dump or a CICS transaction dump rarely give enough information. If your site uses ABEND-AID, find out how to get a full SYSMDUMP. To generate a SYSMDUMP, supply a //SYSMDUMP card with the UNIT=3420 TAPE or non-IDRC 3480 cartridge.

What to do When You Call In

Tell us about your organization and yourself

- Your customer ID or organization name.
- Your name.

Your environment

- The release and interface of SAMS:Compress that you are running.
- The operating system and release you are running (i.e., MVS 4.2).
- Any changes made to your system between the time that SAMS:Compress worked and when it didn't work.

The problem

- Has the problem occurred before?
- Can you recreate the problem?
- The exact sequence of events that led to the error.

Collect the documents that describe the problem

- The JCL, sysouts, and console log from all jobs involved.
- The module name, message numbers and complete text of any SAMS:Compress messages that were issued.
- The message numbers and complete text of any system messages that were issued (i.e., from the JES2 log).
- The abend code, module, and offset, if there was an abend.

What Happens to Reported Problems?

Once a problem is reported to Customer Support, it is inserted into our Product Maintenance Request (PMR) tracking system. When a resolution to the problem is found — identified as a PMR fix — it is tested and certified before it is made available to users. Of course, users who have reported the problem will receive the PMR fix as soon as it becomes available. PMR fixes are mailed to other users when requested. This usually occurs after the user reports having symptoms similar to a problem previously reported and fixed. Included with each PMR fix is a detailed description of the problem, and a description of the fix, as well as the fix itself.

Glossary

byte value	The value of the binary number contained in one eight-bit byte. For example, the byte value for the character A is 11000001, the byte value for COBOL LOW-VALUES is 00000000, and the byte value for the character 9 is 11111001.
compression ratio	The measure of the amount of storage space saved by compressing a data set. The ratio is expressed by the following percentage: $100 * \frac{x-y}{x}$ where: x = average record length in bytes before compression y = average record length in bytes after compression
condition group	A group of RDL specifications, enclosed in parentheses, whose applicability to a particular record depends upon finding an equality between a test value coded within the condition group and the value contained in a field of the record currently in process.
condition group series	Two or more adjacently coded, mutually exclusive condition groups separated by one or more spaces (blanks). The first condition group for which an equality is found between the field currently tested and the coded condition group test value applies to the record. The remaining condition groups in the series are skipped for the current record.
expansion overhead	The number of CPU cycles necessary to execute the code required for expansion of a compressed record.
FDT	The File Descriptor Table. The FDT contains all of the control information necessary to compress and expand a particular file. There must be an FDT associated with each file that is compressed and expanded.
field length descriptor	The portion of an RDL specification using custom compression which indicates the length, in bytes, of the field currently defined.
field type code	The portion of an RDL specification which indicates the algorithm that is employed by the SAMS:Compress system to compress and expand the field currently defined.
file number	A value between 0 and 31, inclusive, passed as a parameter to SAMS:Compress subroutines,

identifying the FDT associated with the subroutine call. The file number corresponds to the xx of a TABLxx DD statement defining the FDT in the JCL invoking the job.

IFP

See “internal field pointer.”

implement

Implementing SAMS:Compress is a multi-step process which places a file or data base under the control of SAMS:Compress. The process begins after the SAMS:Compress product and its interfaces, if any, have been installed, and ends with the successful completion of the initial compression.

Implementation steps include identifying the file or data base to the Interactive Dialog (or through SAMS:Compress utilities), analyzing the file or data base, testing compression, evaluating the test compression results, fine-tuning RDL (and repeating the testing and evaluating steps), and converting the file or data base.

internal field pointer

A pointer, maintained by the SAMS:Compress system, which points to the current displacement within the record that is processed. For each record, the value in the internal field pointer is initially zero, corresponding to the first byte of the record. As each field is processed, the internal field pointer is increased by the length of the field, to the position of the next field that is processed. Users can explicitly control the value of the internal field pointer by specifying the Position Function.

length descriptor

See “field length descriptor.”

PDS

A partitioned data set, as defined by IBM Data Management.

**post-compression
bit code**

A variable-length bit string to which a byte value from the uncompressed data set is converted as a result of compression by a type C RDL specification. The most frequently occurring byte value maps to the shortest (fewest number of bits) post-compression bit code. The least frequently occurring byte value maps to the longest post-compression bit code.

prepass

The process by which the SAMS:Compress system reads a file (or a portion of a file) before it is compressed, to collect statistical information, and creates the File Descriptor Table. The File Prepass Utility must be executed for a particular file before it is compressed for the first time.

processing overhead	The number of CPU cycles required to execute the code to perform a SAMS:Compress system function; e.g., to compress, expand, puffup, or puffdown a field or a record.
RDL	The Record Definition Language provided by the SAMS:Compress system for users to select alternative algorithms for compression and expansion of fields and records.
RDL specification	A unit of RDL coding which defines one SAMS:Compress field within a record.
RDW	The Record Descriptor Word, defined by IBM Data Management as a four-byte field prefixing variable-length records. The two high-order bytes contain the actual length of the record as a binary integer.
redundancy check byte	A byte appended by the SAMS:Compress system to each compressed record, representing the logical sum of the bits in the uncompressed record. This value is calculated again upon expansion of the record and compared with the redundancy check byte to ensure data integrity.
redundancy checking	The process of recalculating the logical sum of the bits in an expanded record and comparing the result with the redundancy check byte appended to the compressed record. An equality ensures data integrity.
repetition factor	A value, preceding one or more RDL specifications enclosed by single quotes, indicating the number of times to repeat that sequence of RDL specifications.
repetition group	One or more RDL specifications, enclosed by single quotes and preceded by a repetition factor.
repetition indicator	A post-compression bit code indicating that the preceding post-compression bit code occurs repeatedly in sequence within the compressed record.
SCB	See “SAMS:Compress Control Block.”
SAMS:Compress Control Block	A user-supplied formatted main storage area used by the SAMS:Compress system when processing FDTs in load module format.
SAMS:Compress field	A series of contiguous bytes within an uncompressed record, defined with a single RDL specification as having similar compression characteristics.
type code	See “field type code.”

Variable Symbol	An internal SAMS:Compress register which the user can set using type V, VP or VZ RDL specifications. The register is used to contain the length of a variable-length portion of a record, when this value is determined from information contained in the record.
VS	The coded representation of the Variable Symbol in the field length indicator or repetition factor of an RDL specification.

Index

A

ABEND-AID dump	325
ACB Gen	50, 64
Accessing the FDT	129
ACSMETH	153, 156
Alignment Bytes	167
Allocating GENLIB and FDTLIB	13
AMP Parameter	127
Analyze a Data Base	50
Analyze the Data Base	43, 60
ANS COBOL	149
Application Program Link-Edit with the Subroutines	141
Application Program Requirements	39, 145
Assemble and Link the Program Modules	144
Assemble the DBD Macros for the Data Base	64
Assembler H	28, 38, 144
Assembler Language	129 - 130, 133 - 135, 137, 139, 146 - 147
Authority Code	16
Authorizing a User for the Interactive Dialog	24

B

Batch	110
Best Possible Compression	50
BEST TEST	54
Binary	172, 175
Binary Integer Data	173
Binary Length Indicator	167
Binary Zeros	167
Bit Aligned	170, 175
Bit Switches	172
BLKSIZE Parameter	126
Boundaries of Fields	163
BYPASS Parameter	46, 51
Byte Value	327

C

C1 field type	166, 183
C2 field type	166, 183
C3 field type	166, 183
Call the Subroutines	130 - 138

Call to Subroutine CLOSE	138
Call to Subroutine EXPAND	134
Call to Subroutine PUFFDOWN	136
Call to Subroutine PUFFUP	135
Call to Subroutine SHRINK	133
Callable Shrink Subroutine for CICS	
Install Checklist	143
Character Data	166
Character Frequency Tables	166
Check Byte Mismatch	169
Checklist	5
CICS	36
CICS Macro Library	38, 144
CICS Processing Program Table	7, 143
CICS Program Load Table	7, 143
CICS Requirements	39, 145
CICS Source Library	38, 144
CICS Tables	7, 39, 143, 145
CICS transaction dump	325
CISC/VS	142
CLIST Library	19
CLOSE	128, 132, 138
COBOL	110, 129, 133 - 135, 137, 139, 146, 168, 170
COBOL Application Programs	114, 140
COBOL Command Level	146
COBOL Macro Level	146
COBOL Users	114
Company Name	16
Compressed Record	148
Compressed Record Address	133 - 134, 136, 138
Compression Controls	52 - 55
Compression Percentage	51
Compression Ratio	327
Compression Statistics	118, 121
Compression Test Report	51, 54
Concatenating the Interactive Dialog	21
Condition Group	164, 327
Condition Group Series	327
Conditional Groups	176
Confirmation of a fix	325
Contents of the Installation Tape	8
Control Fields	169
Control File	6, 54, 61
Deleting an Authorized User From	24
Updating the User Data Information in	24
Control File I/O Error Panel	70 - 71
Control Maintenance Menu	22, 78
Control the Size of a Compressed Segment	55
Converting COBOL Programs to Use VARISAM Subroutines	149

Converting FDTs to Load Modules	113
CPU Cycles	162
CPU Overhead	51
CR parameter	148
CRA	133 - 134, 136, 138, 140
CSA	36
Currency Unit	23
Current DFP Level	22
Custom RDL	53
Customer Support	
contacting	325
CWA	143

D

DASD Charge/Mo	23
DASD Purchase Cost	23
Data Base Analysis	60
Data Base Implementation Menu	63
Data Base Title	43, 61
Data Compression Simulation	59
Data Division	149
Data Extraction JCL Generation Panel	100, 102 - 103
Data Integrity	124, 127
Data Sample Extract Parameters	50
Data Set Allocation Error Panel	72 - 73
DBD Data Deletion Confirmation Panel	108
DBD Data Deletion Confirmation Scr	64
DBD Generation	49, 63
DBD Generation Screen	63
DBD Generation Service	57
DBD Macros	49, 63
DBD Macros for the Data Base	50
DBD Module	50, 64
DBDLIB	60
DBDLIB Name	43, 60
DDN	152
Default Condition Group	178
Default DASD Unit	22
Default JCL for the Compression and Expansion Utilities	125 - 126
Default RDL	54
Default RDL Specifications	112, 185
Default Tape Unit	22
Define Default Data Sets Panel	84
Defining Compressed Records in COBOL Application	140
Delete Data Base from Control File Panel	107
Delete Data Base From Control File Screen	64
Delete Obsolete Data Base Information	56, 64
Deleting	

a Data Base from the Control File	64
Deleting an Authorized User from the Control File	24
Deleting an IMS Data Base from the Control File	56
DFSRESLB	33
DLI	47
DSECT	139
DSORG Parameter	126
Dump	325
ABEND-AID, inappropriateness of	325
CICS transaction	325
SYSMDUMP	325
Duplicate Byte Values	167

E

EBCDIC	171
Encrypted	110, 113
Environment Division	149
Exempt a Field from Compression	170
EXPAND	128, 134, 137, 139 - 140
Expansion Overhead	327
Expected Values	163, 171
Express	6
Generating the Program Modules	31
Implementing without the Interactive Dialog	205
Loading the Library	30
Express Features	59
Express File Expansion Utility	127
EXTRACT Parameter	46, 51
Extract Parameters	50

F

Fastpath Utility Support	34
FD Data Record	139
FDT	113, 133 - 134, 136, 138, 146, 162, 182, 327
FDT Deletion Confirmation Panel	77
FDT in Load Module Format	129
FDT Modules	39, 50, 145
FDT Names and a Fullword in CWA	143
FDT Regeneration Confirmation Panel	76
FDT Share	49
FDT Sharing	52
FDT Support Services Menu	75
FDT User Parameters	55
FDTLIB	13, 19
FDTLOADR Utility	117
FDTNAMES	36, 39, 143 - 144
FDTs	36

FDTs Maintained in Sequential Data Set Forma	129
Field	163
Field Length Descriptor	165 - 166, 327
Field Lengths	164
Field Type Code	163, 327
Field Type Codes	165
Field Types	166
Fields Exempted from Compression	169
File Compression Utility	118 - 123
File Descriptor Table	113, 146, 162, 182
File Descriptor Table Identifier	133 - 134, 136, 138
File Expansion Utility	124
File Number	327
File Prepass Utility	113 - 114
File Section	139 - 141, 149
File Utility Programs	112 - 116
Filler Characters	174
Fillers	167
Fine-tuning the RDL	51
Fix	
confirmation of	325
Fixed Expected Values	163
Fixed Offset	170
fixed-length	41, 59, 164
Floating Point	170, 172, 175
FN 153	

G

GA field code	167
GA field type	183
Garbage Data	167
Generate a Stored Data Report	50
Generate JCL for Testing	51
Generate Modified DBD Source Macros	63
Generate the DBD Macros for the Data Base	49, 63
Generate the FDTs for the Data Base	49
GENLIB	13, 49, 63
Glossary	327, 329
Guide to Correct RDL Specifications	183 - 184

H

HARDCOPY Screen	61
HDAM	44, 61
HELP	43, 61
Hexadecimal Format	171
HIDAM	44, 61
HISAM	44, 61

HSSR	47
HSSR Facility	51
Huffman Algorithm	166

I

IFP 164, 179, 328	
Implementation Steps Outline	42
Implementing Express for an IMS Data Base	60 - 63
Implementing SAMS:Compress	328
IMS Data Base Administrator	41, 59
IMS Data Base Analysis Panel	86
IMS Data Base Implementation Menu	104
IMS Data Extraction JCL Generation (Full Function) Panel	101
IMS Data Sample Extract Parameter Maintenance Panel	91
IMS Database Segment Count Maintenance Panel	96
IMS Database Stored Data Report Generation Panel	88
IMS Database Title Maintenance Panel	90
IMS DBD Analysis Duplicate ddname Confirmati	87
IMS DBD Generation Panel	106
IMS JCL Generation Menu	97
IMS JCL Generation Menu (Full Function)	98 - 99
IMS Parallel	59
IMS Parallel DL/I	41
IMS Segment Selection Panel	105
IMS Support Services Menu	85
IMS User Data Set Name Maintenance Panel	95
IMS User FDT Parameter Maintenance Panel	94
IMS User Parameter Maintenance Menu	89
IMS.VS.RESLIB	33
IMS/VS Edit/Compression User Exit	59
IMS/VS Macro Library	28
IMS/VS Macros	26, 32
IMS/VS RESLIB	29
IMSHRINK	187
IMSHRINK Module	27
IMSHRINK Operation	200
IMSPASS	187, 190, 193
IMSPASS Execution JCL	197
IMSPASS Parameter File	195
IMSPASS Utility Module	26
Incorporating Subroutine Calls in Existing Application Programs	139
Initialize Interactive Dialog Control File	13
Insert Tally of Actual Length	167
Install	5
Callable SHRINK Subroutines for CICS	142
Install Checklist	5 - 7
Install Tape	8
Install the Interactive Dialog	10 - 24

Installation	5
Installation Name	22
Installation Site	16
Installation Tape	8
Installing	
Callable Subroutines for use with CICS	35 - 39
Interactive Dialog	6, 187
Listing Authorized Users	24
Interactive Dialog Control File	15, 19, 44, 49, 61, 64
Interactive Dialog Primary Services Menu	60
Internal Field Pointer	164, 179, 328
Invalid Data	170, 175
Invalid Fields	170
IOTYPE	153
ISAM Files	149, 185
ISAM records	110
ISPF	187
ISPF Version 2	6
ISPF Version 2.0	41, 59
ISPLIB Library	17

J

JCL Implications for Existing Application Programs	130
JCL procedures	8
JCL to Extract the Sample Data	51

K

KEY	154, 157
Key Fields	41, 59, 170
KEYLEN Parameter	126

L

L field type	167, 183
Left-justified Zoned Numeric Data	174
Length Descriptor	328
Length Indicator	167
Link-Editing VARISAM Subroutines with User Application Programs	157 - 159
Linkage Section	149
Linking Subroutines with Application	141
Listing	
Interactive Dialog Authorized Users	24
Load Module	113
Load source modules	143
Loading the Interactive Dialog Lib	12
Loading the JCL Library	8 - 9

Loading the Load Library	10
LRECL	119
LRECL for Compressed Data Sets	118
LRECL Parameter	126

M

MA field type	168, 183
Maintain Installation Control Parameters Panel	79 - 80
Maintenance Request See PMR	
Match Key Fields	112, 170
Match Keys	132
MB field type	168, 183
MIN SIZE	55
Minimum Size	55
Model Application Program Link-Edit with Subroutines	142
Modes of Use SAMS:Compress/2	111
Modifying the ISPSLIB	17
Modifying the RDL	52
Multiple Record Formats	164
Multipunched	171, 177
Multistage Batch Application Programs	132
Multitasking	110
MVS Environment	41, 59

N

N field type	169, 184
Naming Conventions	17
Night Batch CPU Charge	23
Night On-Line CPU Charge	23
Night Time EXCP Charge	23
Non-Graphic Data	171
Non-graphic Values	177
NOTRUNC	141
Numeric Characters	163

O

Online Environment	132
OS/VS2	41, 59
OUTDB Data Set	197
Overview	5

P

Packed Decimal Data	170, 173
PAD Facility	55
PAD LEN	55
PAD TYP	55
Padding	55
Parameters for Subroutine EXPAND	148
Parameters for Subroutine PUFFDOWN	138
Parameters for Subroutine PUFFUP	136
PARMFL	195
Pattern Matching	168
PCB Number	48
PD field type	170, 184
PDS	328
Performance	162
Permanently Unused Fields	167
PL/I	129, 146
PL/I F	147
PL/I Optimizer	133 - 135, 137, 139, 147
PMR	142, 158
fix	326
tracking	326
Post-Compression Bit Code	328
PPT	39, 144
Prepass	112, 328
Prepass Statistics	113, 115
Primary Services Menu	74
Prime Batch CPU Charge	23
Prime On-Line CPU Charge	23
Prime Time EXCP Charge	23
Procedure Division	149
Procedure Division Input/Output Statements	149
Processing Overhead	162, 329
Processing Overhead of PUFFUP	135
Processing Program Table	39, 145
Processor Speed	23
Product Maintenance Request	142, 158
See PMR	
Program Load Table	39, 145
PSB	48
PSBLIB	48
PUFFDOWN	128, 131, 136, 139
PUFFUP	128, 131, 135, 139 - 140

R

RC parameter	133 - 134, 136, 138, 148
RCD-AREA	154 - 156

RDL	109, 161, 329
RDL Condition Groups	176 - 178
RDL Defaults	185 - 186
RDL Field Type Descriptions	166 - 174
RDL for a Data Base Segment	53
RDL for the Compressible Area	54
RDL for the Segment Key	54
RDL Maintenance Selection List Panel	92
RDL Position Function	179 - 181
RDL Repetition Groups	175
RDL Specification	329
RDL Syntax	164 - 165
RDL User Parameters	54
RDW	167, 329
Re-entrant	113, 128
RECDEF Data Set	165
RECDEF DD Statement	114, 185
RECFM Parameter	126
Record Definition Language	109, 112, 161, 164
Record Descriptor Word	167
Record Keys	132
Recreating a Lost or Damaged FDT	66
Redundancy Check Byte	120, 124, 127, 329
Redundancy Check Byte Calculation	169
Redundancy Checking	329
Reload the Data Base	50, 64
Reorganization/Restructure	57
Repetition Factor	175, 329
Repetition Group	329
Repetition Indicator	329
Repetition Factor	164
Repetition Group	164
Repetition Indicator	117
Report Printing Option Panel	69
Reports	207
Restrictions on RDL Use	182
Return Code	134, 136, 138, 148
Right-justified Zoned Numeric Data	174
RKP Parameter	127

S

S field type	171, 184
Sample Data Extract Control Parameters	46
SAMS:Compress	
Assembling and Linking Program Modules	39
Changing a Data Base Under Control of	56 - 57
Generating Program Modules	37
Identify the Data Base to	49

Interactive Dialog Panel	67
Panels	65
SAMS:Compress Control Block	129, 148, 329
SAMS:Compress Control Blocks	146
SAMS:Compress Field	329
SAMS:Compress for IMS	6
Generating the Program Modules	26
Installing	25 - 29
SAMS:Compress for IMS Express	6
Installing	30 - 33
SAMS:Compress for IMS Library	25
SAMS:Compress Level	22
SAMS:Compress/2	109
FDT Names and a Fullword in the CWA	143
File Utility Programs	112 - 116
in CICS Application Programs	142
Subroutines Under CICS	145
Using the Subroutines	128
SCB	129, 146, 148, 329
Secured SAMS:Compress Services	22
Security	110, 113
SEGCC Macro	201
SEGCC Macros	49
SEGM Macro	49, 57, 63, 201
Sequential Files	185
Set of Expected Values	171
SFEP\$000	67
SFEP\$UUI	68
SFEP1ANA	86
SFEP1DBD	106
SFEP1DDC	108
SFEP1DEL	107
SFEP1DUPD	87
SFEP1FDT	105
SFEP1IMN	104
SFEP1JCL	97
SFEP1JD1	100
SFEP1JDA	101
SFEP1JEF	102 - 103
SFEP1JFF	98 - 99
SFEP1MNU	85
SFEP1RPT	88
SFEP1UDS	95
SFEP1UDT	90
SFEP1UFP	94
SFEP1UPM	89
SFEP1URL	92
SFEP1URP	93
SFEP1USC	96

SFEP8DDC	77
SFEP8FGC	76
SFEP8MNU	75
SFEP9CMP	79 - 80
SFEP9DSN	84
SFEP9MNU	78
SFEP9UDC	83
SFEP9UID	81
SFEP9USR	82
SFEPAFIO	70 - 71
SFEPALER	72 - 73
SFEPHCPY	69
SFEPMENU	74
SFS	52
SHARE=YES	53
Sharing an FDT	52
SHR=SFS	52
SHRIMS	19
SHRINK elements	145
SHRINK subroutine	128, 131, 137, 139, 141
SHRINK.IMEXP.SOURCE	31
SHRINK.IMS.SOURCE	26, 29
SHRINK.LOAD	29, 38, 144
SHRINK.SHR2.SOURCE	38, 144
SHRINK/IMS Release 3.4	29
SHRINK/IMS Release 3.5	29
SHRKCICS	39, 144 - 145
SHRKEXPD	39, 141, 145
SHRKSCBS	39, 144 - 145
SHRKSTUB	39, 141, 145, 158
SHRNKMOD	39, 144 - 145, 147
Simulated Compression	51
SKIP Parameter	46, 51
SMUII	50
Sort Key Fields	112, 170
Sort Keys	132
Specifying Compression Controls	52 - 55
STAT	152, 154 - 155, 157
Stored Data Report	44, 52, 57, 61
Subroutines	112
of SAMS:Compress/2, using	128
Support Services Menu	60 - 61, 64
Syntax Errors	165
SYSLMOD DD Statement	117
SYSDUMP	325
SYSOUT Class	61
SYSPROC	21

T

TABL00 DD Statement	118
TABLxx DD Statement	129
Teleprocessing	110
Textual Data	163
The PUFFDOWN Process	136
The PUFFUP Process	135
Transferring the JCL from the Tape	8
Trial Compression	123
TSO Logon ID	16
TSO User	6
Type Code	329
Type of Data	163

U

UN field type	172, 184
Uncompressed Fields	112
Uncompressed Record	148
Uncompressed Record Address	133 - 134, 136, 138
Uncompressed Record Length	133 - 134, 148
Undefined Fields	172
Unknown User ID Panel	68
Unload the Data Base	50, 64
Updating the User Data Information in the Control File	24
UR parameter	148
URA	134, 136, 138, 140
URA parameter	133
URL	134, 148
URL parameter	133
Usage Computational-3	170
User Data Maintenance Panel	82
User Deletion Confirmation Panel	83
User Maintenance Menu	81
User Modifications to the Interactive Dialog	17
User RDL Parameter Maintenance Panel	93
Using SAMS:Compress/2 Under CICS	142 - 148
Utility Programs	111

V

V field type	173, 184
Variable Symbol	164, 173, 330
variable-length	41, 59, 164, 170, 175
Variable-length Fields	173
Variable-length Input Records	167
Variable-length ISAM Subroutines	149 - 156

Variable-length records	110
Variably Occurring Field	173
VARISAM	149 - 156
VISCLOSE Subroutine	157
VISDELETE Subroutine	157
VISOPEN Subroutine	152
VISREAD Subroutine	154
VISREWRT Subroutine	156
VISWRITE Subroutine	155
VP field type	173, 184
VS 164, 330	
VS field type	173
VSAM CI Splits	55
VSAM Files	186
VZ field type	173, 184

W

WORD	152, 154 - 157
Working-storage Section	139 - 141, 149

X

X field type	171, 184
------------------------	----------

Z

ZL field type	174, 184
zoned decimal data	173 - 174
ZR field type	174, 184