

# System Programmer's Guide to: z/OS System Logger

Managing, optimizing, and sizing the System Logger environment

How exploiters can get the most out of System Logger

System Logger installation, setup, and usage



Frank Kyne  
Stephen Anania  
Paola Bari  
Gianmario Bescapè  
Jim Grauel  
Hiram Neal  
Andrew Sica  
Bill Stillwell  
David Viguers

**Redbooks**





International Technical Support Organization

**System Programmer's Guide to: z/OS System Logger**

**July 2007**

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page ix.

Archived

**Second Edition (July 2007)**

This edition applies to Version 1, Release 8 of z/OS (product number 5694-A01, 5655-G52).

© Copyright International Business Machines Corporation 2003, 2007. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team that wrote this Redbooks document .....	xi
Become a published author .....	xiii
Comments welcome .....	xiii
<b>Summary of changes</b> .....	xv
July 2007, Second Edition .....	xv
<b>Chapter 1. Introduction to System Logger</b> .....	1
1.1 What is the z/OS System Logger? .....	2
1.1.1 How System Logger is used .....	3
1.1.2 Where System Logger stores its data .....	4
1.2 Assumptions .....	4
1.3 Logger exploiters .....	4
1.4 Division of responsibilities .....	5
<b>Chapter 2. System Logger fundamentals</b> .....	7
2.1 Common System Logger terms .....	8
2.2 System Logger installation and setup tasks .....	10
2.3 LOGR CDS and System Logger policy .....	11
2.3.1 LOGR CDS usage .....	12
2.3.2 The System Logger policy .....	15
2.4 Common System Logger services and reporting .....	16
2.5 Log streams .....	22
2.5.1 CF-Structure based log streams .....	23
2.5.2 DASD-only log streams .....	45
2.5.3 Updating log stream definitions .....	55
2.5.4 Deleting log stream and CF structure definitions .....	57
2.6 Offload processing .....	58
2.6.1 CF-Structure size and offloading .....	63
2.6.2 Staging data set size and offloading .....	63
2.7 Deleting log data .....	64
2.7.1 When log data is marked for deletion .....	64
2.7.2 When is my log data or offload data set physically deleted? .....	65
2.8 System Logger recovery .....	67
2.8.1 Failure independence .....	67
2.8.2 Failure recovery .....	69
2.8.3 Other recovery processes and errors .....	72
2.9 An introduction to SMF88 records .....	77
2.10 Recommended service .....	77
2.10.1 Collecting documentation for problem diagnosis .....	77
2.10.2 Recommended maintenance policy .....	80
<b>Chapter 3. DFSMStvs and Logger</b> .....	81
3.1 Function description .....	82
3.1.1 Log stream usage .....	85

3.1.2	Criticality/persistence of data	87
3.1.3	Log streams sizing	87
3.2	Definitions	90
3.2.1	Subsystem definitions	90
3.2.2	Logger definitions	92
3.2.3	Coupling Facility log streams	92
3.2.4	Security definitions	104
3.3	Operational considerations	105
3.3.1	Can you stop/start the use of Logger	105
3.3.2	Defining log streams dynamically	105
3.3.3	Displaying log stream information	106
3.3.4	Monitoring for offload conditions	107
3.4	Recovery	107
3.4.1	Missing data in the log streams	107
3.4.2	Log stream full condition	108
3.4.3	CF failures	109
3.4.4	System Logger failure	110
3.4.5	Subsystem failure	112
3.5	Performance	112
	<b>Chapter 4. IMS Common Queue Server and the System Logger</b>	<b>113</b>
4.1	Functional description	114
4.1.1	Overview of shared queues in the IMSplex	114
4.1.2	How CQS uses the System Logger	118
4.1.3	System Logger offload processing	122
4.1.4	Offload data set deletion	123
4.1.5	Importance of the log stream data	124
4.2	Defining the System Logger in the IMSplex	124
4.2.1	Subsystem (CQS) definitions	124
4.2.2	System Logger definitions	124
4.2.3	Security definitions	130
4.3	CQS and System Logger operational considerations	130
4.3.1	Controlling the use of System Logger	130
4.3.2	Defining log streams dynamically	130
4.3.3	Displaying log stream information	130
4.3.4	Monitoring for offload conditions	131
4.4	CQS and System Logger error recovery considerations	132
4.4.1	Non-zero return and reason codes from the System Logger	132
4.4.2	Important messages	133
4.5	CQS and System Logger performance considerations	134
4.5.1	Structure checkpoints	134
4.5.2	Structure checkpoint recommendations	135
4.5.3	Staging data sets	135
4.5.4	Staging data set recommendations	135
4.5.5	Measurements	136
4.5.6	Measurement environment	136
4.5.7	Utilities and tools	137
4.5.8	Measurement methodology and reports	138
4.5.9	Measurement results	138
4.5.10	Summary of measurement results	142
4.5.11	Additional considerations	142
	<b>Chapter 5. CICS and System Logger</b>	<b>145</b>

5.1	Function description	146
5.1.1	Log stream usage	148
5.1.2	Criticality/persistence of data	150
5.2	Definitions	151
5.2.1	Subsystem definitions	151
5.2.2	System Logger definitions	154
5.3	Operational considerations	169
5.3.1	RACF authorizations	169
5.3.2	Data set lifetime	171
5.3.3	HSM considerations	173
5.3.4	Disaster recovery	174
5.3.5	Log stream sizing	175
5.3.6	Performance	187
<b>Chapter 6. Other Logger exploiters</b>		<b>209</b>
6.1	APPC protected conversations	210
6.1.1	Functional description	210
6.1.2	Definition	212
6.1.3	Operational considerations	214
6.1.4	Recovery	215
6.1.5	Performance	215
6.2	Logrec	216
6.2.1	Functional description	216
6.2.2	Definition	217
6.2.3	Operational considerations	219
6.2.4	Recovery	221
6.2.5	Performance	221
6.3	OPERLOG	221
6.3.1	Functional description	222
6.3.2	Definition	222
6.3.3	Operational considerations	226
6.3.4	Recovery	228
6.3.5	Performance	228
6.4	Resource Recovery Services	228
6.4.1	Functional description	228
6.4.2	Definition	231
6.4.3	Operational considerations	240
6.4.4	Recovery	241
6.5	System Automation for OS/390	242
6.5.1	Functional description	242
6.5.2	Definition	243
6.5.3	Operational considerations	245
6.5.4	Recovery	245
6.5.5	Performance	246
6.6	WebSphere Application Server	246
6.6.1	Functional description	246
6.6.2	Definition	247
6.6.3	Operational consideration	251
6.6.4	Recovery	251
6.6.5	Performance	252
<b>Chapter 7. Logger operations</b>		<b>253</b>
7.1	Reporting - System Logger status	254

7.1.1	Display Logger command . . . . .	254
7.1.2	LIST Option on IXCMIAPU . . . . .	256
7.1.3	Display GRS command. . . . .	258
7.1.4	How to identify log streams with problems / damaged log stream . . . . .	258
7.2	Offload monitoring . . . . .	261
7.3	Stopping/Starting Logger Address Space . . . . .	262
7.4	Handling Directory Full Condition . . . . .	262
7.5	Handling of Couple Data Sets. . . . .	263
7.6	HSM considerations . . . . .	264
7.7	Deletion of a log stream with valid data in interim storage . . . . .	265
7.8	Structure rebuilding . . . . .	266
7.9	Tools . . . . .	270
7.9.1	DUMP command parmlib member . . . . .	270
7.9.2	SMF88 . . . . .	270
7.9.3	IXGRPT1. . . . .	271
7.9.4	LOGR Couple Data Set Audit Tool . . . . .	272
<b>Chapter 8.</b>	<b>System Logger performance and tuning . . . . .</b>	<b>273</b>
8.1	Introduction . . . . .	274
8.2	Estimating log stream sizes . . . . .	274
8.2.1	Sizing offload data sets. . . . .	274
8.2.2	Sizing interim storage . . . . .	275
8.3	System Logger duplexing of interim storage. . . . .	281
8.3.1	CF-Structure log stream . . . . .	282
8.3.2	DASD-only log streams. . . . .	283
8.3.3	Sizing duplex copy of CF log streams. . . . .	283
8.4	Setting HIGHOFFLOAD and LOWOFFLOAD thresholds . . . . .	284
8.4.1	High and low offload thresholds for funnel-type log streams . . . . .	284
8.4.2	High and low offload thresholds for active log streams. . . . .	284
8.5	Other performance recommendations . . . . .	285
8.5.1	Which CF log streams to place in the same structure. . . . .	285
8.5.2	System Logger DASD performance . . . . .	286
8.5.3	Indicators of potential problems . . . . .	286
8.6	Using System Logger performance data. . . . .	286
8.6.1	RMF: Analysis of the CF structure service times . . . . .	287
8.6.2	RMF: Workload Activity Reports . . . . .	289
8.6.3	SMF Type 88 records and IXGRPT1 program . . . . .	291
8.6.4	IXGRPT1 Field Summary and IXGSMF88 Cross Reference . . . . .	293
8.6.5	DFSORT jobs to format SMF 88 records . . . . .	297
8.6.6	Sample spreadsheet tool to analyze SMF 88 records . . . . .	302
<b>Chapter 9.</b>	<b>Disaster Recovery considerations . . . . .</b>	<b>305</b>
9.1	Overview . . . . .	306
9.2	Providing time consistent data . . . . .	306
9.2.1	Peer to Peer Remote Copy - PPRC . . . . .	306
9.2.2	Extended Remote Copy - XRC . . . . .	307
<b>Appendix A.</b>	<b>Rebuilding the LOGR Policy . . . . .</b>	<b>309</b>
	JCL used to rebuild the LOGR policy. . . . .	310
	LOGRPOL CLIST. . . . .	311
<b>Appendix B.</b>	<b>Additional material . . . . .</b>	<b>313</b>
	Locating the Web material . . . . .	313
	Using the Web material . . . . .	313



How to use the Web material .....	313
<b>Related publications</b> .....	315
IBM Redbooks .....	315
Other publications .....	315
Online resources .....	316
How to get IBM Redbooks .....	316
<b>Index</b> .....	317

Archived

Archived

# Notices

This information was developed for products and services offered in the U.S.A.

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

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

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

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

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

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

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

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

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

## COPYRIGHT LICENSE:

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

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	IMS™	Redpaper™
DB2®	MQSeries®	Redbooks (logo)  ®
ESCON®	MVS™	RMF™
FICON®	NetView®	VTAM®
GDPS®	OS/390®	WebSphere®
Geographically Dispersed Parallel Sysplex™	Parallel Sysplex®	z/OS®
IBM®	RACF®	z9®
	Redbooks®	

The following terms are trademarks of other companies:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

# Preface

The z/OS® System Logger is a function provided by the operating system to exploiters running on z/OS. The number of exploiters of this component is increasing, as is its importance in relation to system performance and availability. This book provides system programmers with a solid understanding of the System Logger component and guidance about how it should be set up for optimum performance with each of the exploiters.

## The team that wrote this Redbooks document

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Frank Kyne** is a Senior IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM® classes worldwide in all areas of Parallel Sysplex®. Before joining the ITSO five years ago, Frank worked in IBM Global Services in Ireland as an MVS™ Systems Programmer.

**Stephen Anania** is an Advisory Programmer in Poughkeepsie, NY. He has 20 years of experience in z/OS. He joined IBM in 1983 after graduating from the University of Pennsylvania with a bachelor's degree in Mathematics. Stephen spent the first ten years at IBM in JES2 development, and then worked for several years in MVS System Test. He transferred to his current position in pre-sales technical marketing support in the beginning of 2002.

**Paola Bari** is an Advisory Programmer in Poughkeepsie, NY. She has 20 years of experience as a Systems Programmer in OS/390®, z/OS and Parallel Sysplex, and a few years of experience in WebSphere® MQSeries® and WebSphere Application Server.

**Gianmario Bescape** is an Advisory Software Support with IBM GTS in Italy. He has been working in IMS™ Technical Support group in Milan for about 20 years, and is currently a member of the EMEA IMS second level support team. His experience includes support in solution of problems and in the implementation of IMS Shared Queues environments for large customers.

**Jim Grauel** was an IBM Senior Technical Staff Member. He joined IBM in 1965 as a Hardware Customer Engineer, in St. Louis, Missouri. In 1972 he transferred to Software as a Program Support Representative, supporting OS R21.7 and CICS® (Release 2.3). In 1980 he transferred to the CICS Support Center in San Jose, California, and later to Raleigh in 1984 and has been working as Level 2 Support in the CICS Support Center.

**Hiram Neal** is an Advisory Software Engineer with the IMS Performance Group at the Silicon Valley Laboratory in San Jose, CA. He has been a member of the IMS Performance team for 9 years, conducting pre-release IMS performance tests and customer support for IMS versions 6-10.

**Andrew Sica** is a Software Engineer. He holds a degree in Computer Science from Western New England College and has four years of experience working in System Logger development and service.

**Bill Stillwell** was a Senior Consulting I/T Specialist and has been providing technical support and consulting services to IMS customers as a member of the Dallas Systems Center for 21

years. During that time, he developed expertise in application and database design, IMS performance, Fast Path, data sharing, shared queues, planning for IMS Parallel Sysplex exploitation and migration, DBRC, and database control (DBCTL). He also develops and teaches IBM Education and Training courses, and is a regular speaker at the annual IMS Technical Conferences in the United States and Europe. He recently co-authored a book for IMS Version 8 implementation and several IMS Parallel Sysplex Redbooks® documents.

**David Viguers** is a member of the IBM IMS development team at Silicon Valley Lab working with performance, test, development, and customer support. Dave also develops and delivers education on IMS Parallel Sysplex. He has spent over 35 years working with IMS in various functions and organizations.

Thanks to the following people for their contributions to this project:

Rich Conway  
International Technical Support Organization, Poughkeepsie Center

Bob Haimowitz  
International Technical Support Organization, Poughkeepsie Center

Mario Bezzi  
IBM Poughkeepsie

Nicole Burgess  
IBM Poughkeepsie

Juliet Candee  
IBM Poughkeepsie

Tarun Chopra  
IBM Poughkeepsie

Andrew Currier  
IBM Poughkeepsie

Evan Haruta  
IBM Poughkeepsie

Rick Kready  
IBM Poughkeepsie

Curt Jews  
IBM Washington Systems Center

Robert Miller Jr.  
IBM Poughkeepsie

Bruce Naylor  
IBM Silicon Valley Lab

Tom Rankin  
IBM Poughkeepsie

Stephen Warren  
IBM Poughkeepsie

Peter Redmond  
IBM Poughkeepsie

Doug Zobre  
IBM Poughkeepsie

Marty Moroch  
IBM Corporation USA

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks document dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an Internet note to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYJ Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

Archived



# Summary of changes

This section describes the technical changes made in this edition of the book. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes  
for SG24-6898-01  
for System Programmer's Guide to: z/OS System Logger  
as created or updated on March 29, 2012.

## July 2007, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

### **New information**

- ▶ Information has been added to Chapter 4, "IMS Common Queue Server and the System Logger" on page 113, reflecting the results of a performance test using modern DASD subsystems with Logger staging data sets in a remote copy environment.

Archived



# Introduction to System Logger

This chapter provides information about what the System Logger component of z/OS is and why it was developed. It also introduces the components and products that exploit Logger and describes the topics in this book that are useful to MVS Systems Programmers and Subsystem Systems Programmers.

# 1.1 What is the z/OS System Logger?

System Logger is an MVS component that provides a logging facility for applications running in a single-system or multi-system sysplex. The advantage of using System Logger is that the responsibility for tasks such as saving the log data (with the requested persistence), retrieving the data (potentially from any system in the sysplex), archiving the data, and expiring the data is removed from the creator of the log records. In addition, Logger provides the ability to have a single, merged, log, containing log data from multiple instances of an application within the sysplex.

Log data managed by the System Logger may reside in processor storage, in a Coupling Facility structure, on DASD, or potentially on tape. However, regardless of where System Logger is currently storing a given log record, from the point of view of the exploiter, all the log records are kept in a single file that is a limited size. The location of the data, and the migration of that data from one level to another, is transparent to the application and is managed completely by System Logger, with the objective of providing optimal performance while maintaining the integrity of the data. This is shown in Figure 1-1.

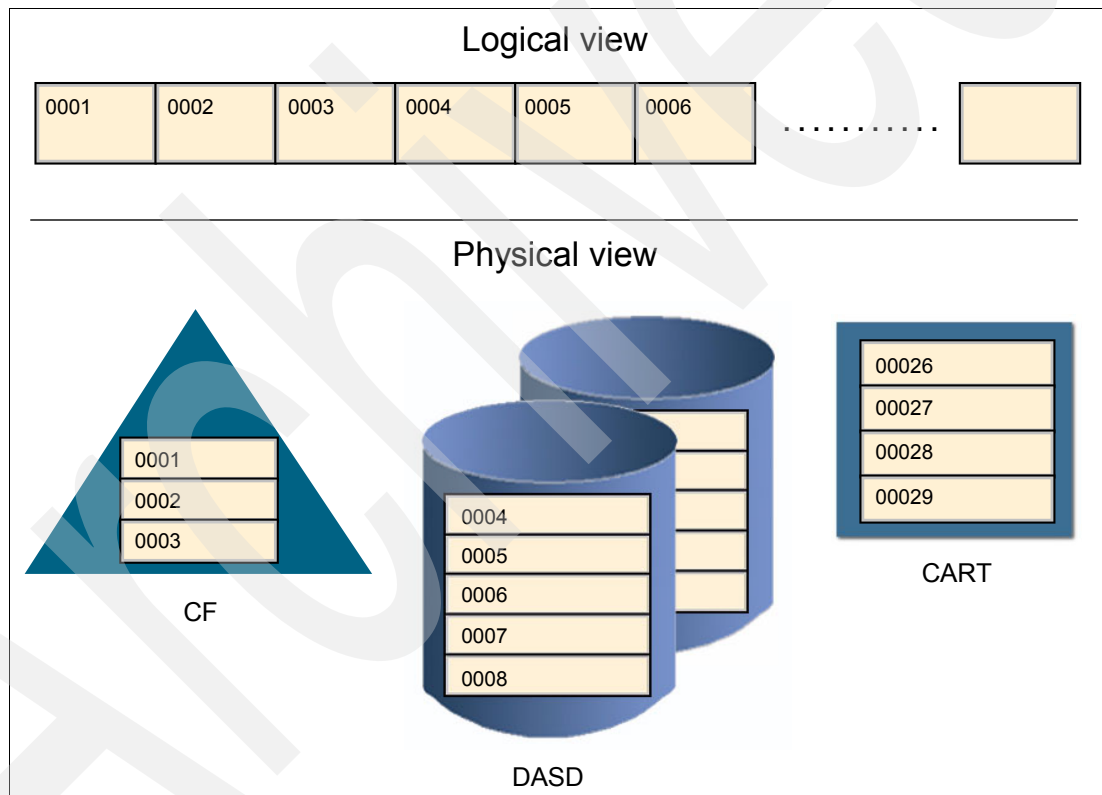


Figure 1-1 Logical and physical views of System Logger-maintained log data

The task of tracking where a specific piece of log data is at any given time is handled by System Logger. Additionally, System Logger will manage the utilization of its storage—as the space in one medium starts filling up (a Coupling Facility structure, for example), Logger will move old data to the next level in the hierarchy.

By providing these capabilities using a standard interface, many applications can obtain the benefits that System Logger provides without having to develop and maintain these features themselves. This results in faster development, more functionality, and better reliability. Enhancements to System Logger, such as support for System Managed CF Structure Duplexing, become available to all System Logger exploiters as soon as they are

implemented in System Logger, rather than having to wait for each exploiter to design, write, and test their own support.

### 1.1.1 How System Logger is used

There are basically two types of users of System Logger. Some exploiters basically use System Logger as an archival facility for log data. These exploiters dump their log data into System Logger and rely on it to manage the archival and expiration of the data from there on. Of course, these exploiters have the ability to subsequently retrieve the data should they need to do so, but their normal mode of operation would be to just give data to System Logger and not look for it back again. An example of this type of exploiter would be the CICS Forward Recovery logs, where CICS stores data away in case a forward recovery is required some time in the future. In this book, we call these exploiters *funnel-type* exploiters.

The other type of exploiter typically uses the data more actively, and explicitly deletes it when it is no longer required. An example of this would be the CICS DFHLOG. CICS stores information in DFHLOG about running transactions, and deletes the records as the transactions complete. We call these types of exploiters *active* exploiters,

As you can imagine, the performance requirements of these exploiters will differ. The exploiters that use Logger primarily to archive data are not particularly concerned about retrieval speeds, whereas an active user of the data will ideally want all the active data to be kept in a high performance location, like a data space or a CF structure. One of the objectives of this part of the book is to help you identify and provide the appropriate levels of service for the various users of System Logger in your installation. Table 1-1 contains a list of the current System Logger exploiters, and shows for each one what type of exploiter it is (funnel-type or active).

Table 1-1 Current System Logger exploiters

Exploiter	Exploiter Type
APPC Protected Conversations	funnel-type
CICS DFHLOG	active
CICS DFHSHUNT	active
CICS user journals	funnel-type
CICS forward recovery logs	funnel-type
CICS log of logs	funnel-type
DFSMStvs IGWLOG	active
DFSMStvs IGWSHUNT	active
IMS Shared Queues	funnel-type
Logrec	funnel-type
OPERLOG	funnel-type
RRS Active log	active
RRS Archive	funnel-type
RRS Resource Manager data	funnel-type
RRS Delayed URs	funnel-type

Exploiter	Exploiter Type
RRS Restart	funnel-type
SA/390 Message Log	funnel-type
SA/390 Workitem History Log	funnel-type
SA/390 Health Checker History	funnel-type
WebSphere Error Log	funnel-type

### 1.1.2 Where System Logger stores its data

When an application passes log data to System Logger, the data can initially be stored on DASD, in what is known as a DASD-only log stream, or it can be stored in a Coupling Facility (CF) in what is known as a CF-Structure log stream. The major differences between these two types of log stream configurations are the storage medium System Logger uses to hold interim log data, and how many systems can use the log stream concurrently:

- ▶ In a CF log stream, interim storage for log data is in CF list structures. This type of log stream supports the ability for exploiters on more than one system to write log data to the same log stream concurrently.
- ▶ In a DASD-only log stream, interim storage for log data is contained in a data space in the z/OS system. The data spaces are associated with the System Logger address space, IXGLOGR. DASD-only log streams can only be used by exploiters on one system at a time.

## 1.2 Assumptions

The primary objective of this book is to provide you with the information you require to effectively set up and manage System Logger and its interactions with its exploiters. We do not assume that you already have System Logger up and running. However, in order to keep the book to a manageable size, there are some assumptions that we have had to make:

- ▶ You have at least a minimal SMS configuration
- ▶ Your system is currently configured as a monoplex or a sysplex

Additionally, we have had to make some assumptions about your experience and skills. If you do not have all the required skills, there are education offerings and other documentation that can help you obtain those skills. The skills we assume that you possess are:

- ▶ You know how to code and implement SMS routines and classes and how to configure SMS
- ▶ You are familiar with at least basic sysplex set up and operations. If you have a Parallel Sysplex, we assume that you are familiar with setting up and maintaining a CFRM policy
- ▶ You understand the installation and customization process for the subsystems that will be exploiting System Logger, at least as it relates to the use of System Logger

## 1.3 Logger exploiters

The System Logger was first made available as part of MVS/ESA V5.2, and was initially used by Operlog, Logrec, and CICS, which used System Logger to replace its own log and journal

files. Since then, as more products introduce sysplex exploitation support, the list of System Logger users has grown. The list now includes:

<b>CICS</b>	CICS use System Logger for its logs (DFHLOG, DFHSHUNT, and DFHLGLOG) and user journals. While each CICS region maintains its own logs, it is possible to create a merged forward recovery journal by specifying that all the CICS regions that update a given file write to the same forward recovery journal.
<b>TVS</b>	DFSMS Transactional VSAM Services for its logs (IGWLOG, IGWSHUNT, and log-of-logs). While each TVS subsystem maintains its set of Undo logs (LOG+SHUNT), it is possible to have the log-of-log log stream and the forward recovery log streams shared with the CICS systems.
<b>IMS</b>	The IMS Shared Message Queue facility uses System Logger to log data created as messages are processed from the shared message queue. The log data is only subsequently required if it becomes necessary to recover either of the Shared Message Queue structures.
<b>RRS</b>	The MVS Resource Recovery Services component of z/OS has a number of different types of log data (up to five), and it uses System Logger to store and manage this data in a manner that allows the data to be accessed from any system in the sysplex.
<b>APPC</b>	To provide resource recovery for protected conversations, APPC/MVS requires the names of local and partner LU logs, and the negotiated sync point capabilities for each local/partner LU pair. This information needs to be available and accurate for successful resynchronization after a failure. To store this information, APPC/MVS uses a System Logger log stream.
<b>Merged Logrec</b>	In a Parallel Sysplex environment, it is possible to have the systems write any Logrec records to a single merged log stream.
<b>OPERLOG</b>	In a Parallel Sysplex environment, you can create a sysplex-wide view of the syslog activities by writing syslog records to a single merged log stream. This provides a view of activity at the sysplex level, which is invaluable when trying to debug multi-system problems.
<b>SA/390</b>	System Automation for OS/390 can write two types of data to a log stream. The first of these is messages produced by the z/OS HealthChecker component of SA/390. The other is produced by the Automation Manager and it is organized across two log streams: the workitem and the message log.
<b>WebSphere</b>	WebSphere Application Server uses a log stream to writes error messages. You can have a log stream per each cloned server or a shared log stream shared across multiple instances of servers.

## 1.4 Division of responsibilities

In order to get the required levels of availability and performance for the sysplex, it is vital that the MVS Systems Programmers have a solid understanding of how System Logger works and the effect of changing Logger and log stream parameters. However, when defining a log stream for an exploiter such as CICS or IMS, information is required that is normally not available to the MVS Systems Programmers.

Similarly, when deciding on the appropriate parameters for a given log stream, it is important to understand the effect that different log stream parameters can have, both on that log stream, and on System Logger in general.

Therefore, we recommend that the MVS Systems Programmers read all the chapters concerning System Logger, including the chapters related to the subsystems (meaning basically to read Chapter 1 through Chapter 8). Equally, we recommend that the Subsystem Systems Programmers read the chapter relating to the subsystem they are managing, for example, the CICS System Programmer should read the CICS chapter, and also the chapters that are dealing with the general concepts of the logger environment, such as:

- ▶ Chapter 2, “System Logger fundamentals” on page 7
- ▶ Chapter 7, “Logger operations” on page 253
- ▶ Chapter 8, “System Logger performance and tuning” on page 273
- ▶ Chapter 9, “Disaster Recovery considerations” on page 305



# System Logger fundamentals

In this chapter we provide information about the System Logger component of z/OS. In order to use System Logger effectively, and to deliver the availability and performance demanded in modern sysplex environments, it is important that you understand the basics of how System Logger works.

This chapter discusses:

- ▶ Common terms
- ▶ System Logger installation and set up tasks
- ▶ LOGR Couple Data Set and System Logger policy
- ▶ Common System Logger services and reporting
- ▶ Log streams (CF-Structure based versus DASD-only)
- ▶ Offload processing
- ▶ Deleting log data
- ▶ Failure independence
- ▶ Introduction to SMF88 data
- ▶ Service recommendations

## 2.1 Common System Logger terms

There are a number of terms used when discussing System Logger that are common across all the exploiters. In order to understand the subsequent discussions in this book, it is important to be familiar with these terms. Figure 2-1 shows how log data managed by System Logger spans multiple storage mediums. This is a useful picture as it helps to introduce several of the terms we will be discussing.

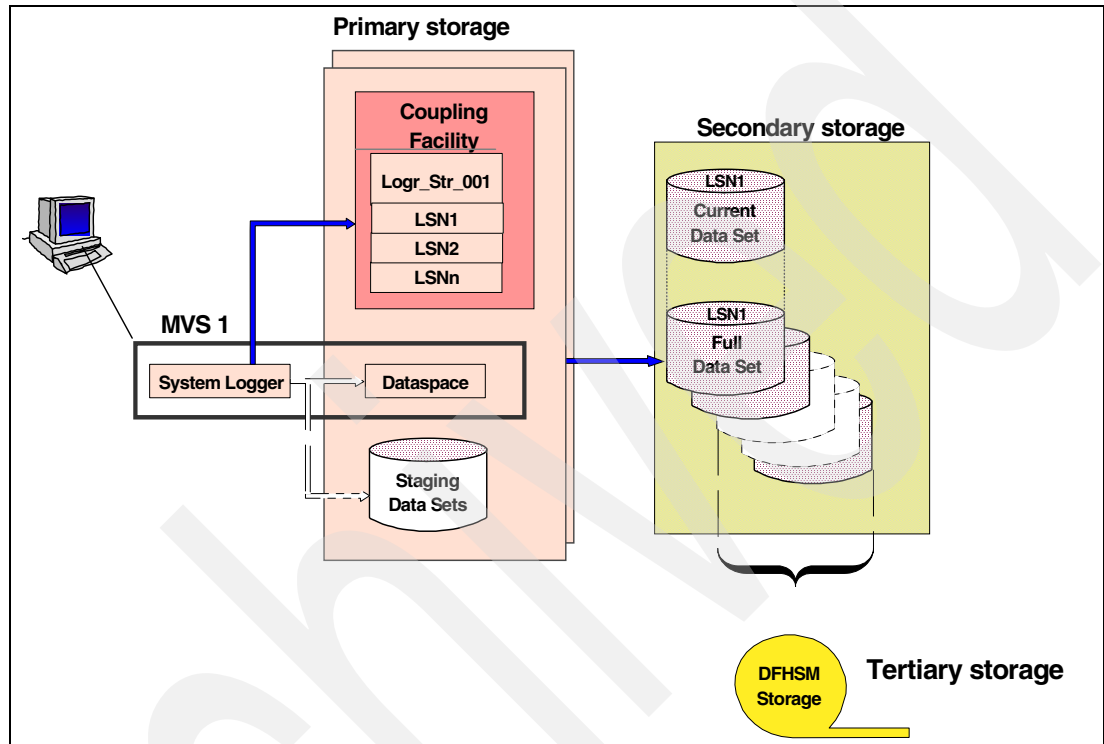


Figure 2-1 System Logger spanning different levels of storage

Figure 2-1 gives you a picture of a system logger configuration. Following are terms that are used to describe system logger elements and processes:

### System Logger

A component of the operating system that provides logging services. This component is commonly referred to as the MVS System Logger, OS/390 System Logger, z/OS System Logger, or simply as System Logger or just Logger. While there are enhancements that have been introduced by specific releases of the operating system, these terms are generally used interchangeably, irrespective of the release or version of the operating system being used.

### Log streams

A sequence of data blocks, with each log stream identified by its own log stream identifier—the log stream name (LSN). Log streams data can span multiple recording media: interim (primary) storage, secondary (DASD based) storage, and tertiary (that is, tape migrated) media.

### Interim Storage

Interim storage is the primary storage used to hold log data that has not yet been offloaded. The interim storage medium used depends on how the log stream has been defined; it may be a Coupling Facility (CF) structure or a staging data set. Log data that is in interim storage is duplexed to prevent against data loss conditions. The data is usually

duplexed to a data space, although log streams residing in a CF structure may optionally be duplexed to a staging data set.

### **LOGR Couple Data Set**

The LOGR Couple Data Set (CDS) holds the System Logger policy information and information about all defined log streams, and must be accessible by all the systems in the sysplex.

Defining the LOGR CDS and the System Logger policy information is one of the first tasks you must complete when preparing to migrate to a System Logger environment. See *z/OS MVS Setting Up a Sysplex*, SA22-7625 for information about the LOGR CDS.

### **CF structure definitions**

All the CF structures used by the System Logger must be defined in advance in the CFRM policy. See *z/OS MVS Setting Up a Sysplex*, SA22-7625, for information about defining the CFRM policy.

### **Log stream definitions**

The log streams are defined using the IXCMIAPU program, and the definitions are stored in the System Logger policy in the LOGR CDS.

### **Offload**

The process of moving valid (not yet deleted) data from interim storage to offload data sets. An offload is generally triggered when the high offload threshold specified for the log stream is reached; however there are other events that will cause an offload to occur. See 2.6, “Offload processing” on page 58 for more information.

### **Offload data sets**

Auxiliary storage on DASD for log streams, sometimes also referred to as log data sets or log stream data sets; they are single extent VSAM linear data sets. When the interim storage for a log stream is filled to its high offload threshold point, the System Logger begins offloading data from interim storage to the offload data sets.

The naming convention for offload data sets is:

<EHLQ or HLQ>.<streamname>.<seq#>

### **Staging data sets**

Interim storage on DASD used to duplex log data that has not yet been offloaded to offload data sets. Staging data sets are required for DASD-only log streams, and are optional for CF-Structure based log streams.

The naming convention for staging data sets differs based on the type of log stream.

For *CF-Structure* based log streams, the naming convention is:

<EHLQ or HLQ>.<streamname>.<system name>

For *DASD-only* log streams, the naming convention is:

<EHLQ or HLQ>.<streamname>.<sysplex name>

### **Log record**

A log record is a single record as produced by the exploiter. It could be something like a line from syslog, for example. Or it could be a before-image created by a data manager before it updates a record in a database.

### **Log block**

Exploiters have a choice when sending data to System Logger. The log data can be sent

every time a new log record is created, or log records can be written to a buffer and subsequently sent to System Logger. In either case, the piece of data that is sent to System Logger is called a *log block*. It is transparent to System Logger whether the log block contains a single log record or many log records.

## 2.2 System Logger installation and setup tasks

There are a number of requirements that must be taken into account when planning for System Logger and any applications that exploit it. Table 2-1 illustrates the steps required to set up System Logger. If you plan on just using DASD-only log streams, only the common setup activities (those on the left side of the table) are necessary; CF-Structure based log stream users will need to complete all the steps listed. These requirements will be discussed further throughout this chapter. For more information about System Logger installation requirements, see *z/OS MVS Setting Up a Sysplex*, SA22-7625.

Table 2-1 System Logger installation tasks

Common Setup Activities	Additional CF-Structure log stream steps
<b>Sysplex Environment</b>	
<ul style="list-style-type: none"> <li>▶ Need to be in a sysplex environment (PLEXCFG keyword in IEASYSxx)</li> <li>▶ Set up sysplex and LOGR Couple Data Sets and System Logger policy</li> <li>▶ Tailor Parmlib members - IEASYSxx, COUPLExx, IEFSSNxx</li> </ul>	<ul style="list-style-type: none"> <li>▶ Need to be in a Parallel Sysplex environment</li> <li>▶ Need a CF</li> <li>▶ Set up CFRM Couple Data Sets and policy</li> </ul>
<b>Log Stream Configuration</b>	
<ul style="list-style-type: none"> <li>▶ Identify types of log streams you will use</li> <li>▶ How many log streams are needed</li> <li>▶ Plan for log stream recovery</li> </ul>	<ul style="list-style-type: none"> <li>▶ Plan structure placement - which log streams will reside in each structure</li> <li>▶ Determine structure sizes</li> <li>▶ Plan for peer connector recovery</li> </ul>
<b>Naming Conventions</b>	
<ul style="list-style-type: none"> <li>▶ Log stream names</li> <li>▶ Log stream DASD data set names</li> <li>▶ Add log stream DASD data set names to GRSRNLxx inclusion list</li> </ul>	<ul style="list-style-type: none"> <li>▶ CF structure names</li> </ul>
<b>DASD Space</b>	
<ul style="list-style-type: none"> <li>▶ Estimate log stream staging and offload data set sizes</li> <li>▶ Allow for DASD expansion - specify additional DSEXTENTS</li> <li>▶ Plan use of retention period for log streams</li> <li>▶ Set up SMS classes for offload and staging data sets</li> <li>▶ If multisystem sysplex: <ul style="list-style-type: none"> <li>– Need shared catalog(s)</li> <li>– Access to DASD volumes</li> <li>– Serialization mechanism</li> <li>– <i>SHAREOPTIONS(3,3)</i></li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▶ Select duplexing method for log stream data - data space or staging data sets</li> </ul>

Common Setup Activities	Additional CF-Structure log stream steps
<b>Security Authorization</b>	
<ul style="list-style-type: none"> <li>▶ IXGLOGR address space needs access to log streams</li> <li>▶ Administrator of LOGR CDS</li> <li>▶ Exploiters of log streams</li> </ul>	<ul style="list-style-type: none"> <li>▶ CFRM CDSs and structures <ul style="list-style-type: none"> <li>– For System Logger address space</li> <li>– Administrator of CFRM CDS</li> <li>– Exploiters of log stream structures</li> </ul> </li> </ul>
<b>Activate SMS</b>	
<b>Activate System Logger</b>	
<b>Establish SMF88 reporting</b>	

**Important:**

- ▶ System Logger requires that the System Managed Storage (SMS) subsystem be *active* on any system using System Logger. This is true even if you do not use SMS to manage offload or staging data sets, since System Logger uses SMS services to allocate them. If SMS is *not active* when you attempt to use System Logger, System Logger will issue messages indicating allocation errors and the application will not be able to use the logging function. If you do not use SMS to manage offload or staging data sets, you can set up a minimal configuration to provide an environment for SMS. However, we recommend using SMS to manage System Logger offload and staging data sets, as it allows greater control over how System Logger uses your DASD storage. For more information about SMS, see *z/OS DFSMS: Implementing System-Managed Storage, SC26-7407*.
- ▶ If you have multiple systems in the sysplex, it is normal for System Logger to require access to offload data sets and staging data sets from multiple systems; for this reason, you *must* specify SHAREOPTIONS(3,3) for those data sets. Incorrect SHAREOPTIONS is a common problem among users of System Logger. If you have the PTF for APAR OW56438 installed, look for messages IXG267I and IXG268I which indicate incorrect SHAREOPTIONS; if you are running without the OW56438 PTF, look for IEC161I and System Logger exploiter messages.

## 2.3 LOGR CDS and System Logger policy

Because the System Logger uses sysplex services, any system that you wish to use System Logger on must either be in monoplex mode (PLEXCFG=MONOPLEX in IEASYSxx) or it must be a member of a multi-system sysplex (PLEXCFG=MULTISYSTEM in IEASYSxx). The requirements to be a member of either type of sysplex are described in *z/OS MVS Setting Up a Sysplex, SA22-7625*. In addition, the System Logger subsystem should be defined in the IEFSSNxx member, using the following statements:

```
SUBSYS SUBNAME(LOGR) INITRTN(IXGSSINT)
```

The System Logger component manages log streams based on policy information that you place in the LOGR CDS. This can be a point of confusion, as you will often see references to the System Logger policy, the LOGR CDS (also called the System Logger CDS), or the System Logger inventory; these are not separate entities. While some other components of z/OS have multiple policies (CFRM and SFM, for example), the LOGR CDS contains the *only* System Logger policy. The System Logger policy contains CF structure definitions (if applicable), log stream definitions, and data describing the current status of all log streams. To understand and effectively manage System Logger, it is important to remember this difference between the LOGR CDS and the other sysplex CDSs.

## 2.3.1 LOGR CDS usage

The LOGR CDS:

- ▶ Must be formatted using the IXCL1DSU utility
- ▶ Contains policy information (more on that in the next topic)
- ▶ Must be accessible by all the systems in the sysplex

Regardless of whether you intend to use CF-Structure based or DASD-only log streams, it is important that you understand how the definition of the LOGR CDS can impact your applications. The parameters you specify can limit the number of log streams, LOGR structures, and offload data sets, as well as the ability of your systems to exploit new System Logger function. It is possible to bring in a new CDS with larger values (at the same LOGR CDS format level or greater), but it is *very* disruptive to move to a CDS with smaller values - therefore, it is important to give proper consideration to the values you will use.

**Restriction:** If you wish to switch to a CDS that is formatted with smaller values, not only do you require a sysplex IPL, you will *also* lose all the information about all the log streams defined in the old CDS, along with all the data in those log streams.

### Formatting the CDS

Format a primary, alternate, and spare LOGR CDS using the IXCL1DSU utility, making sure they are on *different* volumes, and different physical control units if possible, to ensure a copy exists if the primary is lost or otherwise not usable. We recommend you not over-specify these values as it can lead to a much larger than necessary CDS which can result in your IPL time increasing. While there is no sample job in SYS1.SAMPLIB for formatting the LOGR CDS, there is a sample job provided in Appendix B of *z/OS MVS Setting Up a Sysplex*, SA22-7625.

You *may* consider defining the alternate to have slightly higher values than the primary. If the values defined for your primary CDS are too low, you will have an alternate ready to switch in. However if you switch to this alternate for any reason, you will then need to define a new primary, formatted with the same values as the old alternate, that you can switch back to. This methodology should be used judiciously as it can result in the LOGR CDSs gradually getting larger and larger over time, until you end up with CDSs that are much larger than necessary.

The parameters that can be specified when creating a new LOGR CDS are:

#### LSR

The maximum number of *log streams* that can be defined in the System Logger policy that will be stored in this CDS. The default is 1, the minimum is 1, and the maximum is 32767. *Do not* take the default on this parameter or you will be limiting your sysplex to one log stream. You should evaluate the System Logger applications you plan to use and determine the number of log streams needed by each (keeping in mind some applications that run on separate systems may require a single sysplex-wide log stream; others might use separate log streams for each system).

#### LSTRR

The maximum number of *structure names* that can be defined in the System Logger policy. The default is 1, the minimum is 1, and the maximum is 32767. If you plan on using only DASD-only log streams, it is not necessary to specify this parameter. If you are planning on using CF-Structure based log streams, you should determine how many structures are necessary. We will discuss the guidelines for this in 2.5.1, “CF-Structure based log streams” on page 23.

Note that the maximum number of structures that can be defined in a CFRM policy is currently 512, so there is no value in making this number larger than 512 *at a maximum*. In practice, you should not make this number any larger than necessary. Also, system logger supports 255 connected structures at a time per system.

### DSEXTENT

The number of *additional* offload data set directory extents available. The default is 0, the minimum is 0, and the maximum is 99998. Each DSEXTENT (directory extent) goes into a common pool available to any log stream in the sysplex, and is allocated as needed. If you have log streams that will exceed 168 offload data sets (for example, if you retain log data for long periods of time) you should specify this parameter. Log streams start with a base of up to 168 directory entries (that is, 168 offload data sets that can be used, 1 per directory entry); *each* additional DSEXTENT specified will allow the log stream owning the extent to use an additional 168 directory entries.

It is important to keep in mind that a DSEXTENT can only be owned by one log stream. If you have 20 log streams with 169 offload data sets each, you would require at least 20 DSEXTENTs. When all the offload data sets in a DSEXTENT have been physically deleted, it will be returned to the available pool.

**Note:** Staging data sets do not count towards the directory entry limit.

If you decide to specify DSEXTENTs, there are some considerations you should keep in mind when determining how many should be requested. You should specify a value that is at least 15% more than you expect to be in use at any given time. You can monitor usage by:

- ▶ Periodically running a LOGR CDS report using the IXCMIAPU utility. This will show the number of DSEXTENTs in use. Try to keep the number of directory records in use below 85% of the total formatted.
- ▶ Watch for system messages IXG261E and IXG262A, which indicate usage of the offload data set directory is over 85% and 95% respectively.

If you have reached the maximum number of offload data sets and there are no DSEXTENTs in the available pool, System Logger will not be able to create any new offload data sets for that log stream and any offload attempt will fail. Watch for system message IXG301I with a return code of X'08' and reason code of X'085C' indicating this state. When additional DSEXTENTs have been made available for the sysplex, System Logger will retry the offload and issue an ENF 48 indicating that the log stream is available. You may be able to make additional directory space available by deleting log data from the log stream. For more information about deleting log data, see 2.7, "Deleting log data" on page 64.

You *may* be able to set up a retention period and automatic deletion policy to help ensure your log streams will not run out of space; however, be sure to verify that the use of these keywords will not have a negative impact on your System Logger applications. For more information about retention period and autodelete, see "Defining CF-Structure based log streams" on page 33.

### SMDUPLEX

SMDUPLEX specifies whether the LOGR CDS should be formatted at the HBB6603 or HBB7705 level. The default is 0 (NO), and the maximum is 1 (YES). Specifying 1 indicates that System-Managed CF Structure Duplexing is supported, in addition to other enhancements introduced in z/OS 1.3 and 1.4. We discuss CDS format levels further in "LOGR CDS format levels" on page 14.

Ensure you meet all the prerequisite requirements before attempting to use System-Managed Duplexing. For more information about this, see the section entitled "Understanding

System-Managed Duplexing Rebuild Requirements” in *z/OS MVS Setting Up a Sysplex*, SA22-7625.

## LOGR CDS format levels

The addition of the SMDUPLEX parameter to IXCL1DSU for LOGR CDSs in z/OS V1R2 allows the installation to specify what level CDS should be created. If you are not using the newest level LOGR CDS, you will be precluded from exploiting most of the new functionality added to System Logger since z/OS V1R2. The newest LOGR CDS format level as of z/OS V1R4 is HBB7705. To use the HBB7705 LOGR CDS format level, all systems in your sysplex *must* be running z/OS V1R2 or later. Any system running a lower level of z/OS or OS/390 will not be able to access a HBB7705 format CDS, and System Logger will not start on those systems. As we discuss functions that require this format level later in this chapter, we will note the requirement.

**Note:** LOGR CDS format levels do not actually correspond to any particular release in a one-to-one relationship, despite the naming convention (using release FMIDs such as HBB6603 and HBB7705). Instead, the format level only corresponds to the release it was introduced in, as this is the earliest release of OS/390 or z/OS it can be used with.

## Updating the LOGR CDS

If you decide to update the format level of your LOGR CDS, remember that the newly-formatted CDS cannot decrease any of the values defined in the current CDS. To view these values as well as the current format level, you can either run the IXCMIAPU LIST report, or use the D XCF,C,TYPE=LOGR command.

**Attention:** Note that you should never move to a new format level until all systems in the sysplex are stable at the required level of z/OS to support the new format, as System Logger will not start on those systems not meeting the minimum requirement. Attempting to fall back to a lower format level CDS will *corrupt* or *destroy* log data.

There are some steps you should take before moving to a new CDS:

- ▶ Make sure you have already defined your new primary, alternate, and spare CDSs.
- ▶ Update the COUPLExx member in SYS1.PARMLIB to point to the new primary and alternate CDS.

After completing these steps, you have ensured that a new primary and alternate CDSs are ready, and that the COUPLExx member points to them. This is important; if it is necessary to IPL and the COUPLExx member still points to the old CDSs, you will be given the option to switch back to those CDSs, which could result in corruption or loss of System Logger policy data.

To switch to the new LOGR CDS while keeping your policy definitions intact:

- ▶ Make the new primary CDS the current alternate by issuing the following command:  
`SETXCF COUPLE,TYPE=LOGR,ACOUPL=new_primary_cds`
- ▶ Next, promote that data set to be the primary LOGR CDS:  
`SETXCF COUPLE,PSWITCH,TYPE=LOGR`
- ▶ Finally, move in the new alternate:  
`SETXCF COUPLE,TYPE=LOGR,ACOUPL=(new_alternate_cds,volume)`



These steps can be carried out at any time; it is not necessary to quiesce System Logger activity. As a result of taking these steps, you will have new LOGR primary and alternate CDSs and all of your policy definitions (log streams, CF structures, and so on) will be intact.

It is also possible to bring in a new primary and alternate LOGR CDS without retaining your System Logger policy definitions (“clean” copy):

- ▶ Format new primary and alternate LOGR CDSs.
- ▶ Update the COUPLExx member in SYS1.PARMLIB to identify the new primary and alternate CDS to the sysplex.
- ▶ Shut down every system in the sysplex and do a sysplex IPL. After the IPL, you will receive a message asking if the system should use the CDSs listed in the COUPLExx member or the ones used the last time the system was up. You should reply C to indicate that the CDSs listed in the COUPLExx member should be used.

If there are any “failed persistent” structure connections, it may be necessary to force them using the SETXCF FORCE,CON,CONNM=ALL,STRNM=strname command.

**Important:** If you switch to new CDSs using this method, *all the data in the old CDSs will be lost*. Any data in the log streams (including the offload data sets) will be inaccessible. This means that you must define all the log streams again in the new CDSs. It also means that you must manually delete all the offload and staging data sets associated with the log streams that were defined in the old CDSs—if you do not do this, System Logger will not have any knowledge of those data sets and will get duplicate data set errors when it tries to allocate the equivalent data sets for the log streams in the new CDS.

For additional guidance on handling LOGR CDSs, see topic 9.4.8 in *z/OS MVS Setting Up a Sysplex*, SA22-7625, and informational APARs II12375 and II12551.

### 2.3.2 The System Logger policy

The System Logger policy is different than most other z/OS policy definitions in that you can only have one per sysplex. Additionally, whereas most other CDSs only contain static policy statements and some transient information, the LOGR CDS contains persistent information, such as the names of the offload data sets associated with each log stream.

Specifically, the System Logger policy contains:

- ▶ CF structure definitions
- ▶ Log stream definitions
- ▶ Data reflecting the current state of all log streams (connection status, names of staging and offload data sets, High RBA and log block BLOCKIDs for each offload data set, location of the duplex copy of the interim storage, and so on)
- ▶ A list of the connections for every system
- ▶ A count of the number of active systems in the sysplex

Since there is only one System Logger policy per sysplex, you cannot use the DEFINE POLICY NAME parameter in the System Logger policy, nor can you issue the SETXCF START,POLICY command to activate a System Logger policy. The System Logger policy is modified (or updated) using the IXCMIAPU utility or the IXGINVNT macro. As such, it is only necessary to specify the log streams and CF structures that are being *newly* defined, updated or deleted, unlike other policies (such as CFRM) for which you must enter the whole policy. Another difference between the System Logger policy and other policies is that action is taken

*immediately* on the request submitted. For example, if you request that a log stream be defined, it will be done immediately, and the first offload data set will be allocated.

**Note:** One of the more annoying quirks of System Logger's interpretation of input to the IXCMIAPU utility is that if it finds a statement in error, it will ignore all the statements coming after that statement; therefore you should always verify that all the steps in a particular request completed - you can do this by simply checking the job output.

For information about modifying the System Logger policy, see 2.5, "Log streams" on page 22.

### Authorization for IXCMIAPU

Before setting up the System Logger policy with the IXCMIAPU Administrative Data Utility program, you must specify who can use IXCMIAPU:

- ▶ Define a resource profile for the resource name MVSADMIN.LOGR to the SAF FACILITY class. Grant ALTER access to users who must alter or maintain the policy; grant READ access to users who require reports on the policy, but who will not change the policy.
- ▶ If applicable (if you will have CF structures), define a resource profile for the resource name MVSADMIN.XCF.CFRM to the SAF FACILITY class. Grant UPDATE access to users who must define CF structures in the policy.

The users of the IXCMIAPU utility require access to the resources used by the utility. Define the appropriate accesses before users attempt to add, update, delete, or extract a report from the System Logger policy. The LOGR CDS must be activated and available before you add log streams and structure definitions to it. The following access authorities are required to set up the System Logger policy:

- ▶ To use the DEFINE, UPDATE, or DELETE LOGSTREAM statements, ALTER access to the RESOURCE(log\_stream\_name) CLASS(LOGSTRM) profile is required.
- ▶ To use the DEFINE or DELETE STRUCTURE statements, ALTER access to the RESOURCE(IXLSTR.structurename) CLASS(FACILITY) profile is required.
- ▶ To specify a structure name on a LOGSTREAM definition (for example, DEFINE LOGSTREAM(log\_stream\_name) STRUCTNAME(structname)), UPDATE access to the RESOURCE(IXLSTR.structurename) CLASS(FACILITY) profile is required.

## 2.4 Common System Logger services and reporting

System Logger provides a set of services to manage the log data that has been passed to it and to exploit log stream functions. The services allow you to define a log stream, connect to it, and to write, read, and delete data from the log stream. When System Logger encounters an error, the resulting console message will usually include the name of the service that encountered the problem.

### Defining, updating and deleting the log stream - IXGINVNT

The IXGINVNT macro is similar to IXCMIAPU in that it is used to define, maintain and delete CF-Structure and DASD-only log stream information in the System Logger policy.

Using IXGINVNT, an exploiter can:

- ▶ Define and update log stream definitions using REQUEST=DEFINE or REQUEST=UPDATE with TYPE=LOGSTREAM.

- ▶ Define a CF structure that is to be associated with a log stream using REQUEST=DEFINE with TYPE=STRUCTURE.
- ▶ Delete log stream and CF structure definitions using REQUEST=DELETE.

We will discuss log stream and CF structure definition parameters in 2.5, “Log streams” on page 22.

### Connecting to a log stream - IXGCONN

Before you can write, browse, or delete data from a log stream, you must be connected to it. Exploiters use the IXGCONN service to establish a connection to the log stream specified in the request by specifying REQUEST=CONNECT. The service will return a unique connection identifier, called a *STREAMTOKEN*. This is used in subsequent System Logger service requests to identify the exploiter’s connection.

The calling program must also request authorization to the log stream:

- ▶ If AUTH=READ is specified, that application will only be able to issue IXGCONN, IXGBRWSE, and IXGQUERY requests. For AUTH=READ, the caller must have READ access to RESOURCE(log\_stream\_name) in CLASS(LOGSTRM).
- ▶ If AUTH=WRITE is specified, that program can request any System Logger service. For AUTH=WRITE, the caller must have UPDATE access to RESOURCE(log\_stream\_name) in CLASS(LOGSTRM).

An exploiter can connect to the log stream in different ways, for example:

- ▶ *One Connection per Address Space:* Once a program has connected to a log stream, any task running in the same address space shares the connect status and can use the same stream token to request other System Logger services. Any task in the address space can disconnect the entire address space from the log stream by issuing the IXGCONN REQUEST=DISCONNECT call.
- ▶ *One Connection Per Program:* One or more tasks in a single address space can issue IXGCONN REQUEST=CONNECT individually to connect to the same log stream and receive separate stream tokens. Each program must disconnect from the log stream individually.
- ▶ *Multiple Systems Connecting to a Log Stream:* Multiple address spaces on one or more z/OS systems can connect to a single CF log stream, but each one must issue IXGCONN individually to connect to and then disconnect from the log stream. Each one receives a unique stream token; multiple address spaces cannot share a stream token.

The IXGCONN service is also used to disconnect from the log stream by issuing REQUEST=DISCONNECT and specifying the STREAMTOKEN received on connect. This will invalidate the stream token; any future services specifying that token would fail.

When connecting to a log stream, there are additional considerations to be taken into account for each type.

### CF-Structure based log streams

When an application issues IXGCONN to connect to a CF-Structure based log stream:

- ▶ If it is the first connector to the CF structure in this *system*, the System Logger address space will establish a connection to the structure.
- ▶ If it is the first connection to the structure in the *sysplex*, the structure will be allocated.

### **DASD-only log streams**

DASD-only log streams are single system scope; that is, you can only establish connections from one system in the sysplex at a time to a given DASD-only log stream. Any other systems in the sysplex that attempt to connect will fail until all the connections to the log stream from the connected system have been disconnected. However, there can be multiple connections from the *same* system to a DASD-only log stream.

**Note:** For both CF-Structure based (if using staging data sets) and DASD-only log streams, the first connect to the log stream from a given system will result in staging data sets being created.

For more information about using the IXGCONN service, see topic 27.4.5 in *z/OS MVS Assembler Services Guide, SA22-7605*.

### **Writing data to a log stream - IXGWRITE**

Now that the exploiter is connected to the log stream, log records can be written to it. The IXGWRITE service is used to write data to a log stream. The data is stored in interim storage until it can be offloaded (see 2.6, “Offload processing” on page 58 for more information about offload):

- ▶ For *CF-Structure based log streams*, System Logger writes the data to the CF structure space associated with the log stream when an IXGWRITE request is issued. System Logger also duplexes the data to either a System Logger-owned data space or a staging data set. For more information about duplexing log data, see 2.8.1, “Failure independence” on page 67.
- ▶ For *DASD-only log streams*, System Logger writes the data to interim storage in a System Logger-owned data space for the system associated with the log stream and simultaneously to a staging data set on DASD.

**Note:** Log data stored in interim storage is duplexed. For more information, see “Duplexing log data for CF-Structure based log streams” on page 42 and “Duplexing log data for DASD-only log streams” on page 54.

Log blocks can be written either synchronously or asynchronously, depending on the MODE specified on the System Logger IXGWRITE request. Before data can be written out to the log stream, it must be placed in a buffer to form the log block. System Logger has no requirements for the format of the data in log blocks (except that the log blocks cannot be larger than 65532 bytes); it is entirely up to the exploiter as to how the data within the log block is created and used.

For each log block written to the log stream, System Logger will return a unique identifier (the BLOCKID) which can be used on subsequent IXGDELETE and IXGBRWSE requests to search for, delete, read, or set the browse cursor position to that log block. For CF-Structure log streams, the BLOCKID is returned by the CF, ensuring that the BLOCKID is unique for that log stream across the whole sysplex. For DASD-only log streams, because each log stream is only accessed by one system, the BLOCKID is generated by System Logger itself.

The BLOCKID is used when an exploiter wants to retrieve log blocks from System Logger. However, if the data within the log block is going to be used for a time-sensitive task such as recovering a database, the exploiter is responsible for time stamping its log records, and sorting its log records into the correct time order. It cannot assume ascending BLOCKIDs necessarily indicate correspondingly ascending times when the log records were created.

**Note:** It is important to understand that System Logger does *not* guarantee log blocks will be sequenced in the order they were written using the IXGWRITE service. Instead, log blocks are written in the order they are received by System Logger; if you have multiple applications writing to the same log stream, System Logger might not receive them in the same order they were written. System Logger does generate a time stamp in both local and Greenwich mean time (GMT) that indicates the time that the log block was processed. This can be returned to the application by specifying the **TIMESTAMP** parameter.

An imbedded application time stamp will *not* effect the order in which log blocks are written; if an application needs to ensure that log blocks are received into the log stream in the order written, the application can serialize on the log stream.

Let us discuss the logic of how log blocks are written to System Logger; that is, where do log blocks go? Figure 2-2 contains a graphical depiction of System Logger writes. For a CF-Structure based log stream, the log blocks are first written to the CF structure (interim storage) and then duplexed to either staging data sets, a System Logger-owned data space, or another CF structure (depending on several different factors, see “Duplexing log data for CF-Structure based log streams” on page 42 for more information). For a DASD-only log stream, log blocks are written to a System Logger-owned data space and then duplexed to staging data sets (see “Duplexing log data for DASD-only log streams” on page 54).

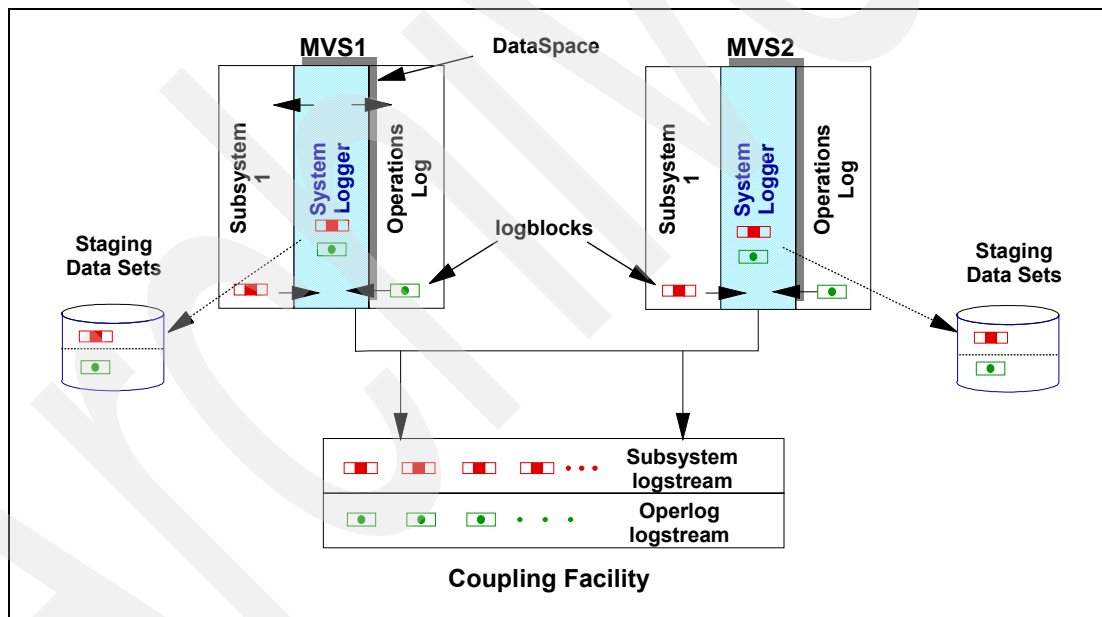


Figure 2-2 Writing to a System Logger log stream

## Browsing and reading log data - IXGBRWSE

Applications can use the IXGBRWSE service to read and browse a log stream for log block information. IXGBRWSE can be used to read consecutive blocks in a log stream or search for and read a specific log block. IXGBRWSE returns the specified log block to the calling program's buffer.

The IXGBRWSE service requires that you have an existing connection. To browse a log stream, exploiters use the following requests:

- ▶ Start a browse session and select the initial cursor position - **REQUEST=START**. This will return a 4-byte browse token; subsequent requests will use this token to identify this browse session. It is possible to have multiple browse sessions running concurrently. Each

browse session also has a browse cursor that points to the current location in the log stream. You can specify the starting position of the browse cursor using one of the following optional keywords:

- **OLDEST** - which is the default, starts the browse session at the oldest (earliest) log block.
  - **YOUNGEST** - starts the browse session at the youngest (latest) log block. Note that the record that is the youngest when the browse session starts might no longer be the youngest record at the end of the browse session because of concurrent write activity to the log stream.
  - **STARTBLOCKID** - specifies that the browse session start at a specified log block identifier. The block identifier for a log block is returned by System Logger when it is written to the log stream (IXGWRITE) in the field specified by the RETBLOCKID parameter.
  - **SEARCH** - specifies that the browse session start at the log block with the specified time stamp. See topic 27.4.7.4 in *z/OS MVS Assembler Services Guide, SA22-7605*, for details on how IXGBRWSE processes time stamps.
- ▶ **REQUEST=RESET** positions the browse cursor to either the oldest or youngest (POSITION=OLDEST or POSITION=YOUNGEST) log block in the log stream.
  - ▶ **REQUEST=READBLOCK** reads a selected log block in the log stream. You identify the block you want to read by either the block identifier (BLOCKID parameter) or the time stamp (SEARCH parameter). The block identifier for a log block is returned by System Logger in the field specified by the RETBLOCKID parameter when the log block is written to the log stream (IXGWRITE).
  - ▶ **REQUEST=READCURSOR** reads the next oldest or youngest log block or blocks in the log stream, depending on the direction specified on the request (DIRECTION=OLDTOYOUNG or YOUNGTOOLD). The block or blocks read also depend on the position of the cursor at the time you issue the request. The *MULTIBLOCK* parameter determines whether one or more blocks will be returned; System Logger will return as many complete log blocks that meet the browse parameter criteria and fit in the caller's buffer.
  - ▶ End a browse session - (**REQUEST=END**). This will result in the browse token being invalidated.

**Note:** Here are some further notes on the IXGBRWSE service:

- ▶ The IXGBRWSE service can be used to browse just active data or all data - even blocks that have been logically deleted (but not physically deleted yet).
- ▶ A browse session may result in an offload data set that has been migrated being recalled; this can cause elongated response times.

## Deleting log data - IXGDELET

Using the IXGDELET service, exploiters can mark some or all of their log blocks in the log stream for deletion. For a CF-Structure log stream, the group of blocks you specify for deletion can reside in both the CF and offload data sets. For a DASD-only log stream, the group of blocks you specify for deletion can reside in both the System Logger-owned data space and offload data sets. The way System Logger processes log data that is marked for deletion depends on whether a retention period and automatic deletion have been specified for a log stream in the LOGR CDS. For more information about how System Logger deletes log data, refer to 2.7, “Deleting log data” on page 64.

Applications use the BLOCKS parameter to specify a range of blocks to be deleted:

- ▶ Either *all* log blocks are deleted or
- ▶ A log block ID is specified, in which case System Logger will delete all log blocks from the log stream that are older than that blockid.

You will note that IXGDELETE only *marks* the log blocks for deletion (that is, they are logically rather than physically deleted). The actual deletion takes place during offload processing, or subsequently when the offload data set is deleted. IXGDELETE is the only way that log blocks get logically deleted—if an IXGDELETE is not issued for a log block, the log block will not be deleted until it has been offloaded and all the log blocks in the offload data set have exceeded their retention period (assuming AUTODELETE(YES) is specified).

### Required access levels

Before using any of these services, you will have to grant access to the classes and resources as shown in Table 2-2.

Return and reason codes for System Logger services can be found in *z/OS MVS Authorized Assembler Services Reference ENF-IXG, SA22-7610*.

Table 2-2 Required SAF accesses for System Logger services

Defining SAF Authorization For System Logger Resources		
System Logger Service	Access Type	SAF class and resource
IXGINVNT REQUEST=DEFINE, TYPE=LOGSTREAM  IXGINVNT REQUEST=UPDATE, TYPE=LOGSTREAM  IXGINVNT REQUEST=DELETE, TYPE=LOGSTREAM	ALTER	RESOURCE(log_stream_name) CLASS(LOGSTRM)
IXGINVNT REQUEST=DEFINE, TYPE=LOGSTREAM, STRUCTNAME=structname	ALTER  UPDATE	RESOURCE(log_stream_name) CLASS(LOGSTRM)  RESOURCE(IXLSTR.structure_name) CLASS(FACILITY)
IXGINVNT REQUEST=DEFINE, TYPE=LOGSTREAM, LIKE=like_log_stream_name	ALTER  UPDATE	RESOURCE(log_stream_name) CLASS(LOGSTRM)  RESOURCE(like_log_stream) CLASS(LOGSTRM)  RESOURCE(IXLSTR.like_structure_name) CLASS(FACILITY)
IXGINVNT REQUEST=DEFINE, TYPE=STRUCTURE  IXGINVNT REQUEST=DELETE, TYPE=STRUCTURE	ALTER	RESOURCE(MVSADMIN.LOGR) CLASS(FACILITY)
IXGCONN REQUEST=CONNECT, AUTH=WRITE	UPDATE	RESOURCE(log_stream_name) CLASS(LOGSTRM)

Defining SAF Authorization For System Logger Resources		
System Logger Service	Access Type	SAF class and resource
IXGINVNT REQUEST=DEFINE, TYPE=LOGSTREAM  IXGINVNT REQUEST=UPDATE, TYPE=LOGSTREAM  IXGINVNT REQUEST=DELETE, TYPE=LOGSTREAM	ALTER	RESOURCE(log_stream_name) CLASS(LOGSTRM)
IXGINVNT REQUEST=DEFINE, TYPE=LOGSTREAM, STRUCTNAME=structname	ALTER  UPDATE	RESOURCE(log_stream_name) CLASS(LOGSTRM)  RESOURCE(IXLSTR.structure_name) CLASS(FACILITY)
IXGCONN REQUEST=CONNECT, AUTH=READ	READ	RESOURCE(log_stream_name) CLASS(LOGSTRM)

## 2.5 Log streams

System Logger exploiters write data to log streams, which can be thought of as simply a collection of data. Log stream storage spans interim storage and offload data sets, where the type of interim storage used depends on the type of log stream.

As discussed previously, the IXCMIAPI utility and optionally IXGINVNT services, are used to define, update, and delete both CF-Structure based and DASD-only log streams. Before choosing which type of log stream (CF-Structure or DASD-only) is best suited to your application, you should consider a couple of attributes of the log data that affect the decision:

- ▶ The location and concurrent activity of writers and readers to a log stream's log data.
- ▶ The volume (data flow) of log stream data.

We will discuss these considerations in the following sections.

It is also important to understand how the parameters specified in the log stream definition will impact storage usage and performance. Some parameters have different meanings or exist only for one type of log stream as indicated in Table 2-3 (note that recommendations still might be different for each type of log stream).

Table 2-3 Parameters by log stream type

Parameters	Definition same for both log stream types?
NAME	same
RMNAME	same
DESCRIPTION	same
DASDONLY	different
STRUCTNAME	only CF-Structure based
MAXBUFSIZE	only DASD-only



Parameters	Definition same for both log stream types?
STG_DUPLEX	only CF-Structure based
DUPLEXMODE	only CF-Structure based
LOGGERDUPLEX	only CF-Structure based
STG_DATACLAS	same
STG_MGMTCLAS	same
STG_STORCLAS	same
STG_SIZE	different
LS_DATACLAS	same
LS_MGMTCLAS	same
LS_STORCLAS	same
LS_SIZE	same
AUTODELETE	same
RETPD	same
HLQ	same
EHLQ	same
HIGHOFFLOAD	different
LOWOFFLOAD	different
LIKE	same
MODEL	same
DIAG	same
OFFLOADRECALL	same

### 2.5.1 CF-Structure based log streams

CF-Structure based log streams can contain data from multiple systems, allowing System Logger applications to merge data from multiple systems throughout the sysplex. These log streams use CF structures for interim storage. Aged log data is then offloaded to offload data sets for more permanent storage. Figure 2-3 on page 24 shows how a CF-Structure based log stream spans the two levels of storage.

When an exploiter writes a log block to a CF-Structure based log stream, System Logger first writes the data to the CF structure; the log block is then duplexed to another storage medium (chosen based on several factors to be discussed later). When the CF structure space allocated for the log stream reaches the installation-defined high threshold, System Logger moves log blocks from the CF structure to offload data sets, so that the CF space for the log stream is available to hold new log blocks. From an exploiter's point of view, the actual location of the log data in the log stream is transparent; however there are configurations that can effect system performance (such as the use of staging data sets).

There are a number of things you must take into account before and after you have decided to use CF-Structure based log streams. If you decide you should use them, then you should

also have a basic understanding of how they work. We spend the rest of this section addressing these points.

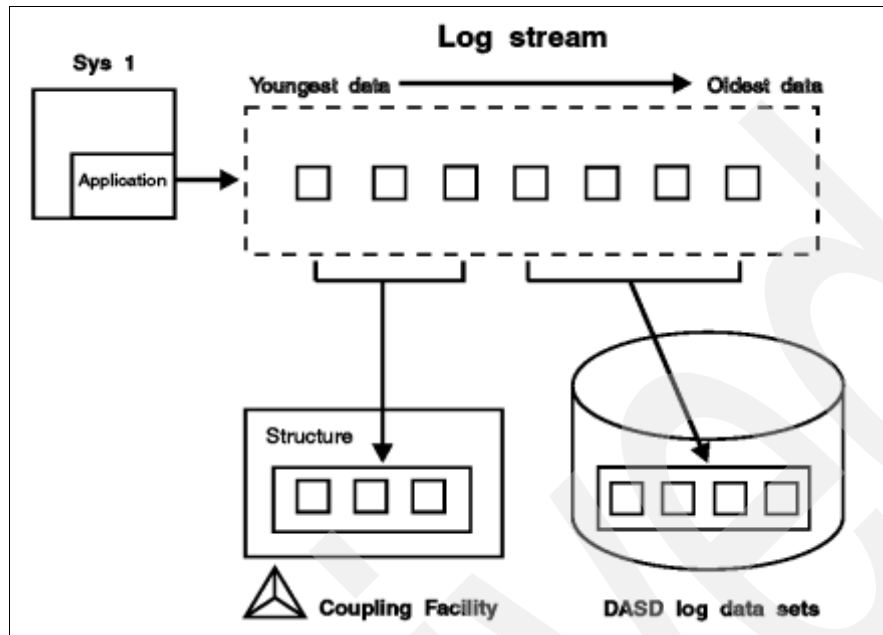


Figure 2-3 Storage levels for a CF-Structure based log stream

### When should my application use CF-Structure based log streams?

Earlier we discussed some factors to be taken into consideration when deciding what type of log stream your application should use:

- ▶ The location and concurrent activity of readers and writers to a log stream's log data.
 

Will there be more than one concurrent log writer or log reader to the log stream from more than one system in the sysplex? For instance, if your System Logger exploiter merges data from multiple systems in the sysplex and you expect the log stream to be connected and be written to by these systems at the same time, CF-Structure based log streams are the only option.
- ▶ The volume (data flow) of log stream data.
 

Will there be large volumes of log data recorded to the log stream? Since DASD-only log streams always use staging data sets, high volume writers of log data may be throttled back by the I/O required to record each log block sequentially to the log stream's staging data sets. Note that even CF-Structure based log streams may use staging data sets for duplexing, depending on the environment System Logger is running in and the parameters specified for the log stream in the System Logger policy.

In addition to these, you should consider any advice given by the exploiter; some exploiters may not recommend DASD-only log streams (APPC/MVS, for example).

**Requirement:** To use a CF-Structure based log stream, you *must* have access to a CF, even for single system scope applications.

### Setting up for and defining CF structures

If you have decided that DASD-only log streams are more appropriate for your applications, skip to 2.5.2, "DASD-only log streams" on page 45, now as the remainder of this section discusses the use of CF-Structure log streams.

If are reading this, you've determined that a CF-Structure based log stream suits the needs of your exploiter, and now you need to set up for them. CF-Structure based log streams require a CF structure, which must be defined in the CFRM policy as well as in the System Logger policy. There are a few questions and concepts that merit further discussion, after which we'll discuss specific definition parameters.

### ***How many CF structures you need***

There are some general recommendations to keep in mind when determining how many CF structures you will need in the sysplex, as well as your log stream configuration:

- ▶ You should always refer to the System Logger exploiter recommendations.
- ▶ It is a good idea to group log streams of similar type (active or funnel-type) and characteristics together, because of the way System Logger allocates space within the CF structures. When you have more than one log stream using a single CF structure, System Logger divides the structure storage equally among the log streams that have at least one connected System Logger application.

For example, if an installation assigns three log streams to a single structure, but only one log stream has a connected application, then that one log stream can use the entire CF structure. When an exploiter connects to the second log stream, System Logger dynamically divides the structure evenly between the two log streams. When another exploiter connects to the third log stream, System Logger allocates each log stream a third of the CF space.

The block size and write rate of the log streams should also be similar (for more information about how this can impact you, see "The entry-to-element ratio" on page 26). Having this information can help you understand and plan how much CF space you have for each log stream and predict how often log stream data will be written to DASD as the CF space becomes filled.

- ▶ Another important consideration is assuring that peer recovery can work where possible. Successful peer recovery requires planning your System Logger configuration such that multiple systems in the sysplex connect to the same CF structure (they don't have to be connected to the same *log stream*, just the same *structure*). If there is no peer connection available to perform recovery for a failed system, recovery is delayed until either the failing system re-IPLs, or another system connects to a log stream in the same CF structure to which the failing system was connected. Where possible, you should *always* plan for peer recovery. For more information, see "Peer and same system log stream recovery" on page 70.
- ▶ In general, try to keep the number of log streams per structure as small as possible; we recommend it be 10 or under. This is related to the **LOGSNUM** parameter (discussed further in "Defining CF-Structure based log streams" on page 33) which specifies the maximum number of log streams for each structure. The main reason you should try to hold to this recommendation is that System Logger connect and log stream recovery processing (which affect the restart time of System Logger applications) has been optimized to provide parallelism at the CF structure level. Therefore, the more structures you use, the greater parallelism you will get during log stream connect and rebuild processing. For example, if you have one structure with four log streams defined to it, System Logger will have to process connect requests to each sequentially. If you divide up those log streams among multiple structures, (for instance, four structures each containing one log stream each), System Logger could process connect requests to each log stream in parallel.
- ▶ System Logger has different numbers of subtasks to carry out different processes. For example, System Logger has one task *per system* to allocate or delete offload data sets. This means that if an allocation or delete request is delayed, all other allocation or delete requests on that system will queue for that task. This is one of the reasons why we

recommend sizing your offload data sets large enough to avoid very frequent allocation of new data sets.

On the other hand, System Logger has one offload thread *per log stream*. So, even if you have ten log streams in the one structure, it would still be possible for all of them to be going through the offload process at the same time.

However, the attribute that should impact your decision of whether to use DASD-only or CF-Structure log streams is the number of tasks to process IXGCONN requests. There is one connect task *per structure*, but only one allocation task *for all staging data sets*. So, let us say you were restarting after a system failure, and all the exploiters are trying to connect back to their log streams. If you are using CF-Structure log streams, and had ten structures, System Logger could process ten connect requests in parallel. However, if all the log streams were DASD-only log streams, System Logger would only process one connect request at a time, obviously elongating the time it would take for all the exploiters to get connected back to their log streams again.

### ***The entry-to-element ratio***

One consideration when planning for CF structure usage and log stream placement is ensuring CF storage is used in an efficient manner. To better understand how System Logger attempts to regulate storage usage requires that we discuss entries and elements.

The important CF structure definition parameters for this discussion are MAXBUFSIZE and AVGBUFSIZE (see “Defining CF-Structure based log streams” on page 33 for details and a complete definition for each). MAXBUFSIZE is used to determine the size of the elements System Logger will use, either 256 bytes (if MAXBUFSIZE is less than or equal to 65276) or 512 bytes (if MAXBUFSIZE is greater than 65276). AVGBUFSIZE (or, after System Logger recalculates this value, the effective AVGBUFSIZE) is then used to determine the entry-to-element ratio so that the ratio is defined as 1 entry per number of elements required to hold an average log block written to the structure. For example, if the element size is 256 bytes (because you specified a MAXBUFSIZE of 65276 or less), and you specify an AVGBUFSIZE of 2560, the initial entry-to-element ratio would be 10:1.

This is a critical point to consider when planning which log streams should reside in the same structure. Entries and elements are created from the pool of storage assigned to a CF structure, based on the entry-to-element ratio currently in use. Let us say, for example, a ratio of 1 entry to 10 elements yields a pool of 1000 entries and 10000 elements. The entries are placed in a pool that can be used by any log stream, and the elements are divided evenly among all the log streams currently connected to the structure, as shown in Figure 2-4.

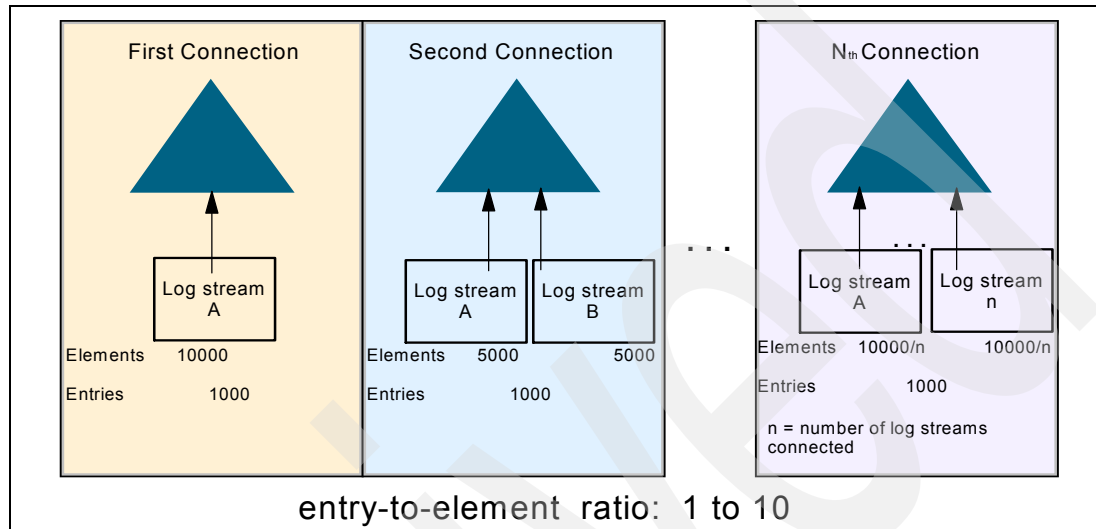


Figure 2-4 Entries and elements divided among connected log streams

Every 30 minutes, System Logger queries the CF structure to determine if the entry-to-element ratio should be dynamically altered. If there is more than a 10% difference between the existing ratio setting and the current *in-use* ratio, System Logger will attempt the alter (at least 20% of the entries and elements need to be in use for System Logger to issue the request, and if more than 90% of either are in use, the request may not be honored). The ratio alteration will result in an *effective* AVGBUFSIZE being used instead of the defined value; the effective AVGBUFSIZE can be seen in the IXCMIAPU LIST STRUCTURE report.

You should note that this is not an exact science as the alter can be effected by temporary spikes in the number of writes or block size; however, over time the value used should closely reflect the average in-use ratio. The exception case is where log streams with significantly different characteristics are defined in the same structure. In this case, the best option is to separate the log streams into different structures.

Let us look at an example of how a log stream writing a different average log block size and a significantly different number of writes per second can impact CF structure efficiency. In Figure 2-5, there are three log streams defined to a structure with an entry-to-element ratio of 1:10.

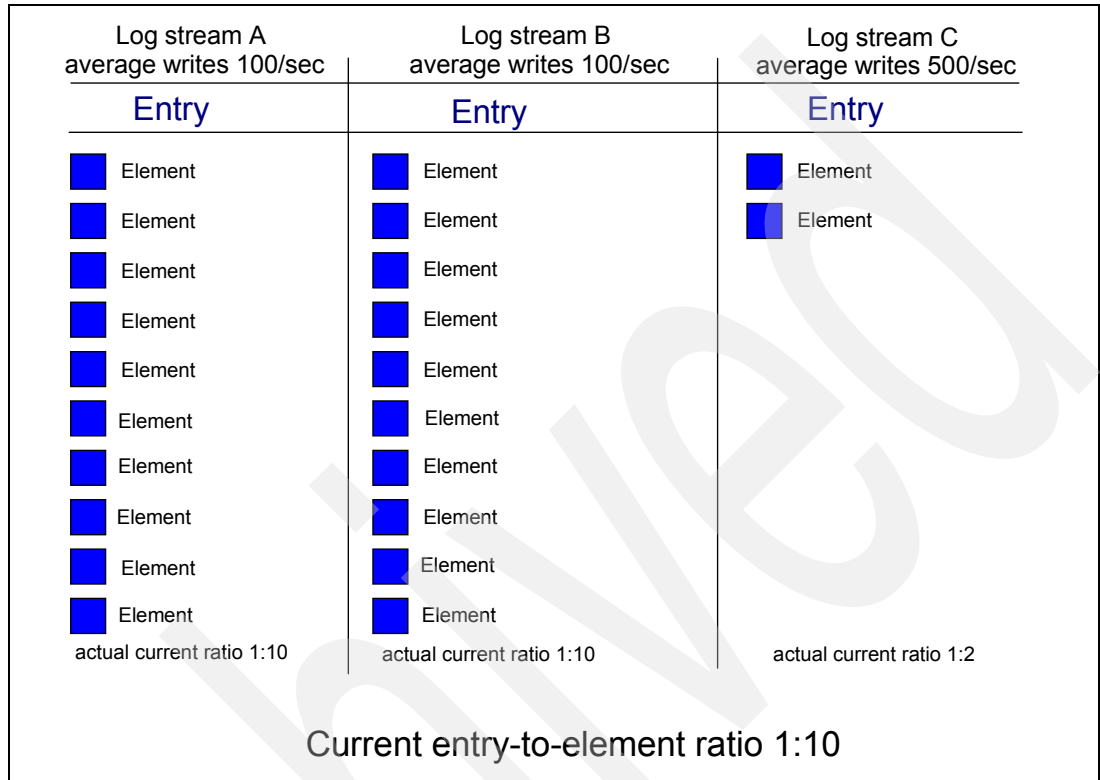


Figure 2-5 Mixing different profile log streams in the same structure

In this example, log streams A and B are similar in both the entry-to-element ratio and number of writes per second; log stream C writes a much smaller log block size and with a greater frequency. Remember, however, they are all given an equal number of elements. Assuming the log streams continue to display the characteristics shown in Figure 2-5 on page 28, let us examine the results as time passes.

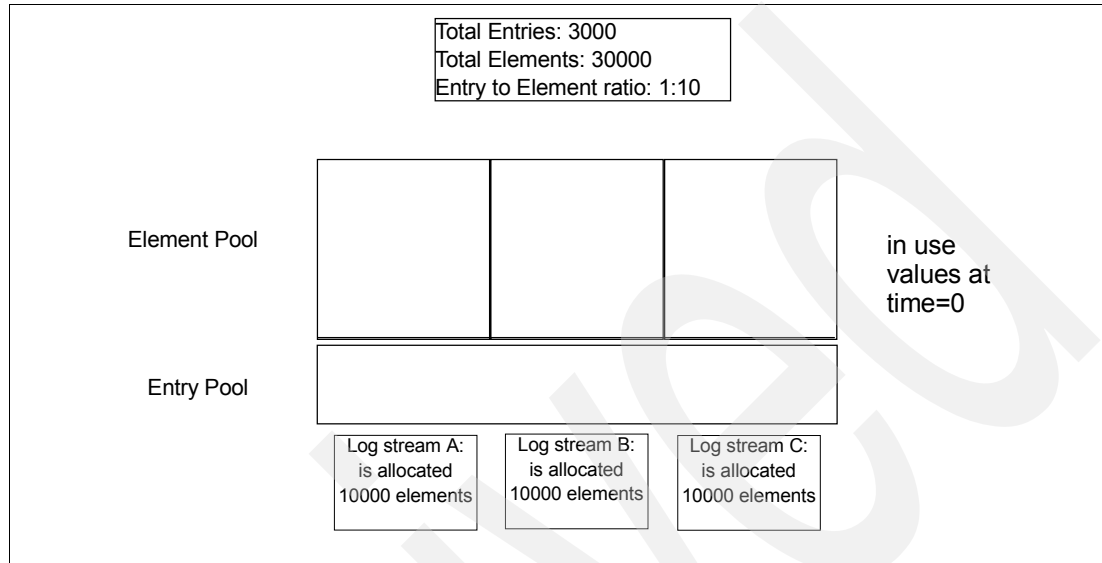


Figure 2-6 Entry/element usage at time=0

At time 0 as shown in Figure 2-6, the three log streams, all defined to the same CF structure, have been allocated an equal number of elements (1/n of the available pool) and they all have access to the pool of 3000 entries.

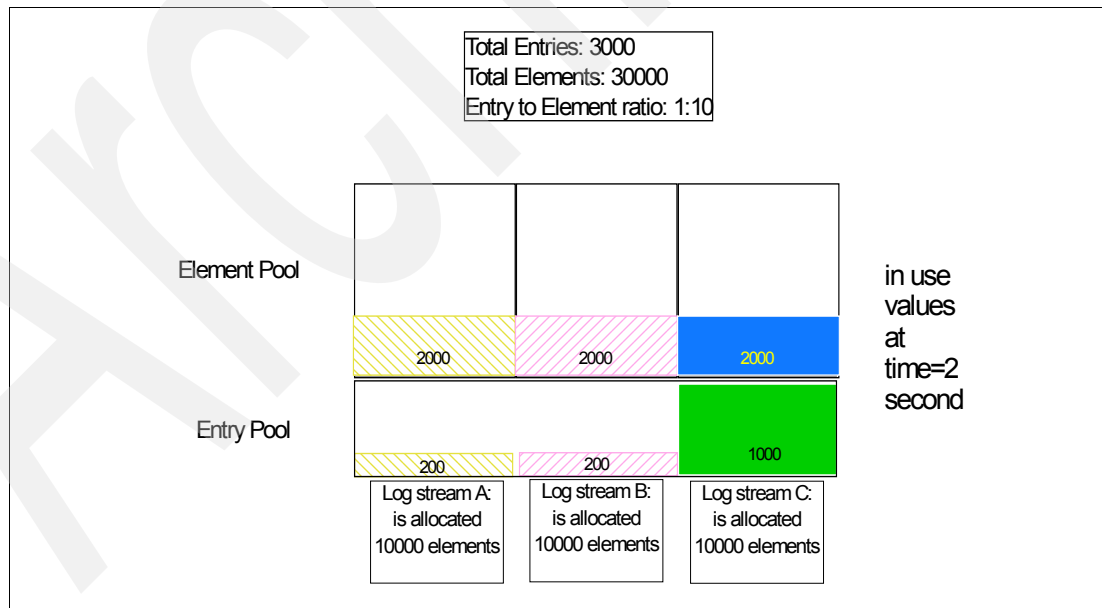


Figure 2-7 Entry/element usage at time=1

Figure 2-7 on page 29 shows at time=2 the beginning of a usage problem. Notice that while log streams A and B are a good pair for this structure, log stream C has already used 33% of the entry pool, but only about 7% of the total element pool.

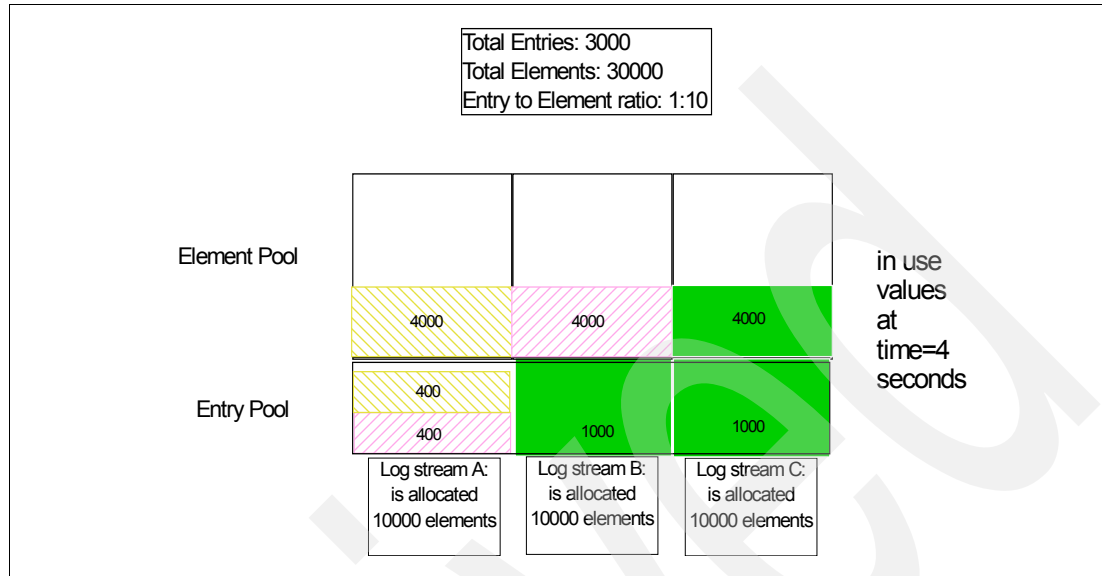


Figure 2-8 Entry/element usage at time=2

Figure 2-8 shows that after 4 seconds, log stream C is using 66% of all entries, far more than its fair share. It will soon run out of entries in the structure, and yet it still has 6000 empty elements. By contrast, log streams A and B are using their fair share and are good candidates to reside in the same structure. In fact, in this example, an offload would be initiated for all the log streams in the structure once 90% of entries have been used. This type of offload moves data from *all* the log streams in the structure and is more disruptive than a normal offload that is kicked off because the HIGHOFFLOAD threshold for a single log stream has been reached. (For more information about offload processing, refer to 2.6, “Offload processing” on page 58.)

Every 30 minutes System Logger would attempt to alter the entry to element ratio in this example, decreasing the pool of available elements and increasing the number of entries for a ratio of around 1 entry to 7 elements (assuming the log stream characteristics in this example stay relatively static). This ratio change would not fix any problems though; assuming we start at time=0 again, it is easy to see that at a future time we will encounter element-full conditions for log streams A and B as they are still using 10 elements per entry and the existing entry-full condition for log stream C will continue to occur. The only way to efficiently resolve this situation would be to move log stream C to a different structure, with a smaller AVGBUFSIZE specified.

### Defining CF structures - CFRM policy

Before using a CF structure defined in the System Logger policy, it must first be defined in the CFRM policy using the IXCMIAPI utility. For a complete list of parameters and an explanation of how to define the CFRM policy, see topic C.2.2 in *z/OS MVS Setting Up a Sysplex*, SA22-7625. In this section we address the parameters critical to System Logger.

It should also be noted that use of any System-Managed Duplexing-related keyword implies that the CFRM CDS has been formatted using the SMDUPLEX(1) keyword.



**Note:** It is a good idea to follow the System Logger application recommendations for CF structure values. You can view recommendations for those applications covered in this book in each of their respective sections.

The CFRM policy keywords that are of interest from a System Logger perspective are:

**NAME**

Must be the same here as specified on the DEFINE STRUCTURE keyword in the System Logger policy.

**SIZE, INITSIZE**

SIZE is the *largest* size the structure can be increased to without updating the CFRM policy. It is specified in 1 KB units.

/NITSIZE is the initial amount of storage to be allocated for the structure in the CF. The number is also specified in units of 1 KB. INITSIZE must be less than or equal to SIZE. The default for INITSIZE is the SIZE value.

You should always refer to the System Logger application recommendations for sizing your CF structures. See the application chapters in this book for recommendations for some of the more common System Logger applications.

IBM provides a Web-based tool to help you determine initial values to use for SIZE and INITSIZE; it is called the CF Sizer and can be found at:

<http://www.ibm.com/servers/eserver/zseries/cfsizer>

Before using the CF Sizer tool, you will need to know some information about the type of log streams that will be connected to the CF structure and the log stream definitions you will use. We will discuss some of the parameters requested by CF Sizer in this chapter. For further information, see topic 9.4.3. in *z/OS MVS Setting Up a Sysplex, SA22-7625*.

**ALLOWAUTOALT**

ALLOWAUTOALT(YES) allows the CF structure size and its entry-to-element ratio to be altered automatically by XES when the CF or the structure is constrained for space.

This parameter should *always* be set to NO (the default) for System Logger CF structures.

As described previously, System Logger has functions for altering the entry-to-element ratio of a structure. Having two independent functions (XES and System Logger) both trying to adjust the ratio can lead to inefficient and unexpected results.

System Logger also manages the usable space within the CF structure and provides for data offloading to offload data sets. If XES were to keep increasing the structure size, it is possible that the log stream would never get to its high offload threshold until the structure had reached its maximum size as defined on the SIZE parameter in the CFRM policy. System Logger is designed to manage the space within the structure size you give it—letting XES adjust the structure size negates this function.

Finally, because System Logger is constantly offloading data from the structure to DASD, is likely that, on average, System Logger CF structures will appear to be only half full. This makes them prime candidates for XES to steal space from should the CF become storage-constrained. Specifying ALLOWAUTOALT(NO) protects the structure from this processing.

Refer to the section entitled “Define the Coupling Facility Structures Attributes in the CFRM Policy Couple Data Set” in *z/OS MVS Setting Up a Sysplex, SA22-7625*, for a discussion of what can happen if you enable ALLOWAUTOALTER for a System Logger structure.

## DUPLEX

The DUPLEX parameter is used to specify if and when the CF structure data will be duplexed. Use of this parameter implies that the sysplex is enabled for System-Managed Duplexing. There are three options for DUPLEX:

- ▶ **ENABLE:** When DUPLEX(ENABLED) is specified for a structure in the CFRM active policy, the system will automatically attempt to initiate duplexing (either user-managed or system-managed) for that structure as soon as it is allocated.
- ▶ **ALLOWED:** When DUPLEX(ALLOWED) is specified for a structure, the structure is eligible for user-managed or System-Managed Duplexing, however the duplexing must be initiated by a connector or by the operator; that is, the structure will not automatically be duplexed by XES.
- ▶ **DISABLED:** Specifies that neither user-managed nor System-Managed Duplexing can be used for the specified structure.

**Note:** Use of the DUPLEX(ENABLED) or DUPLEX(ALLOWED) will impact how System Logger duplexes log data in interim storage. For more information about System Logger duplexing of interim storage, see 2.8.1, “Failure independence” on page 67.

## Defining CF structures - System Logger policy

Now that you have successfully defined your CF structures to the CFRM policy, you can define them to the System Logger policy using either the IXCMIAPU utility or the IXGINVNT macro. The following parameters are used to define CF structures to the System Logger policy:

### STRUCTNAME

Specifies the name of the CF structure you are defining. STRUCTNAME must match the structure name as defined in the CFRM policy, and the structure name as specified on the corresponding log stream definitions.

For CF-Structure based log streams, this is the structure that will be used as interim storage before data is offloaded to offload data sets.

### LOGSNUM

Specifies the maximum number of log streams that can be allocated in the CF structure being defined. *logsnum* must be a value between 0 and 512.

As we discussed previously in “How many CF structures you need” on page 25, the value specified for *logsnum* should ideally be no higher than 10.

### MAXBUFSIZE

Specifies the size, in bytes, of the largest log block that can be written to log streams allocated in this structure. The value for MAXBUFSIZE must be between 1 and 65532 bytes. The default is 65532 bytes.

The MAXBUFSIZE is used to determine what element size will be used; if the value specified is less than or equal to 65276, System Logger will use 256 byte elements. If it is over 65276, 512 byte elements will be used. See “The entry-to-element ratio” on page 26 for more information.

Unless you specifically need a buffer size larger than 65276, we recommend that you specify `MAXBUFSIZE=65276`.

### **AVGBUFSIZE**

Specifies the average size, in bytes, of log blocks written to all the log streams using this CF structure. `AVGBUFSIZE` must be between 1 and the value for `MAXBUFSIZE`. The default value is 1/2 of the `MAXBUFSIZE` value.

System Logger uses the average buffer size to control the initial entry-to-element ratio for the structure. See “The entry-to-element ratio” on page 26 for more information.

**Tip:** Starting with z/OS V1R3, it is no longer necessary to delete and redefine the log streams defined to a CF structure if you wish to move them to another structure. See 2.5.3, “Updating log stream definitions” on page 55, for more information.

### **Defining CF-Structure based log streams**

Now that you have planned your log stream configuration and defined the necessary CF structures, it is time to define the log streams. This is done using the `IXCMIAPU` utility or the `IXGINVNT` service. Let us take a look at the different parameters and how they impact System Logger operations.

#### **NAME**

Specifies the name of the log stream that you want to define. The name can be made up of one or more segments separated by periods, up to the maximum length of 26 characters.

*NAME* is a required parameter with no default. Besides being the name of the log stream, it is used as part of the offload data set and staging data set names:

*offload data set example:* <hlq>.logstreamname.<seq#>  
*staging data set example:* <hlq>.logstreamname.<system>

Note that if the log stream name combined with the HLQ are longer than 33 characters, the data component of the offload and maybe the staging data sets may contain a system-generated low level qualifier, for example:

```
IXGLOGR.A2345678.B2345678.C2345678.A0000000 (Cluster)
IXGLOGR.A2345678.B2345678.C2345678.IE8I36RM (Data)
```

#### **RMNAME**

Specifies the name of the resource manager program associated with the log stream. *RNAME* must be 8 alphanumeric or national (\$,#,or @) characters, padded on the right with blanks if necessary. You must define *RMNAME* in the System Logger policy before the resource manager can connect to the log stream. See the System Logger chapter in *z/OS MVS Assembler Services Guide*, SA22-7605, for information about writing a resource manager program to process a log stream.

#### **DESCRIPTION**

Specifies user-defined data describing the log stream. *DESCRIPTION* must be 16 alphanumeric or national (\$, #,@) characters, underscore (\_) or period (.), padded on the right with blanks if necessary.

#### **DASDONLY**

Specifies whether the log stream being defined is a CF or a DASD-only log stream. This is an optional parameter, with the default being `DASDONLY(NO)`.

Since we are reviewing CF-Structure based log streams in this section, `DASDONLY(NO)` would be used to indicate that we do not want a DASD-only log stream.

**STRUCTNAME**

Specifies the name of the CF structure associated with the log stream being defined.

The CF structure must have already been defined to both the CFRM policy and System Logger policy as discussed in “Setting up for and defining CF structures” on page 24.

**STG\_DUPLEX**

Specifies whether this log stream is a candidate for duplexing to DASD staging data sets.

If you specify STG\_DUPLEX(NO), which is the default, log data for a CF-Structure based log stream will be duplexed in System Logger-owned data spaces, making the data vulnerable to loss if your configuration contains a single point of failure.

If you specify STG\_DUPLEX(YES), log data for a CF log stream will be duplexed in staging data sets when the conditions defined by the DUPLEXMODE parameter are met. This method ensures that log data is protected from a system or CF failure.

For more information about duplexing of log data by System Logger, refer to 2.8.1, “Failure independence” on page 67.

**Note:** Even if you do not plan on using staging data sets, we recommend that you plan for them as there are some failure scenarios under which they would still be used to maintain System Logger’s failure independence. If you have installed the PTF for APAR OA03001 you can specify STG\_DATACLAS, STG\_MGMTCLAS, STG\_STORCLAS and STG\_SIZE parameters even with STG\_DUPLEX(NO). Without the PTFs installed, you cannot specify any of those parameters with STG\_DUPLEX(NO)—in this case, an ACS routine is necessary to associate SMS classes with the staging data sets.

**DUPLEXMODE**

Specifies the conditions under which the log data for a CF log stream should be duplexed in DASD staging data sets.

If you specify DUPLEXMODE(COND), which is the default, the log data will be duplexed in staging data sets only if a system's connection to the CF-Structure based log stream contains a single point of failure and is therefore vulnerable to permanent log data loss.

If you specify DUPLEXMODE(UNCOND), the log data for the CF-Structure based log stream will be duplexed in staging data sets, regardless of whether the connection is failure independent.

**LOGGERDUPLEX**

Specifies whether Logger will continue to provide its own log data duplexing if the structure containing the log stream is being duplexed using System-Managed Duplexing and the two structure instances are failure-isolated from each other.

- ▶ **LOGGERDUPLEX(UNCOND)** indicates that System Logger should use its own duplexing of the log data regardless of whether System-Managed Duplexing is being used for the associated structure.
- ▶ **LOGGERDUPLEX(COND)** indicates that System Logger will only use its own duplexing if the two structure instances are in the same failure domain. If the two instances are failure-isolated from each other, System Logger will not duplex the log data to either a staging data set or a data space.

**CDS requirement:** The active primary TYPE=LOGR CDS in the sysplex must be formatted at a HBB7705 or higher level in order to support the use of the LOGGERDUPLEX keyword. Otherwise, the define request will fail. See “LOGR CDS format levels” on page 14 for more information.

### **STG\_DATACLAS**

Specifies the name of the SMS data class that will be used when allocating the DASD staging data sets for this log stream.

If you specify STG\_DATACLAS(NO\_STG\_DATACLAS), which is the default, the data class is defined through SMS ACS routine processing. An SMS value specified on the STG\_DATACLAS parameter, including NO\_STG\_DATACLAS, always overrides one specified on a model log stream used on the LIKE parameter.

Whether it is preferable to have the DATACLAS association in the LOGR policy, or in the SMS ACS routines depends on your installation software management rules.

Either way, you need to provide a DATACLAS where SHAREOPTIONS (3,3) has been specified for the allocation of the staging data sets. SHAREOPTIONS (3,3) is required to allow you to fully share the data sets across multiple systems within the GRS complex. If your system is running in a monoplex configuration, SHAREOPTION (1,3), which is the default on the dynamic allocation call, is allowed.

See *z/OS DFSMS: Using Data Sets*, SC26-7410, for more information about SMS.

**Important:** Do not change the Control Interval Size attributes for staging data sets to be anything other than 4096. System Logger requires the Control Interval Size for staging data sets be set to 4096. If staging data sets are defined with characteristics that result in a Control Interval Size other than 4096, System Logger will not use them. Operations involving staging data sets defined with a Control Interval Size other than 4096 will fail to complete successfully.

### **STG\_MGMTCLAS**

Specifies the name of the SMS management class used when allocating the staging data sets for this log stream.

If you specify STG\_MGMTCLAS(NO\_STG\_MGMTCLAS), which is the default, the management class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the STG\_MGMTCLAS parameter, including NO\_STG\_MGMTCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS management classes, see Chapter 5 in *z/OS DFSMSdftp Storage Administration Reference*, SC26-7402.

### **STG\_STORCLAS**

Specifies the name of the SMS storage class used when allocating the DASD staging data sets for this log stream.

If you specify STG\_STORCLAS(NO\_STG\_STORCLAS), which is the default, the storage class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the STG\_STORCLAS parameter, including NO\_STG\_STORCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS storage classes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402, Chapter 6.

### **STG\_SIZE**

Specifies the size, in 4 KB blocks, of the DASD staging data set for the log stream being defined. (Note that the size of the CF structures is specified in 1 KB blocks.)

When specified, this value will be used to specify a space allocation quantity on the log stream staging data set allocation request. It will override any size characteristics specified in the specified data class (via STG\_DATACLAS) or a data class that gets assigned via a DFSMS ACS routine.

If you omit STG\_SIZE for a CF-Structure based log stream, System Logger does one of the following, in the order listed, to allocate space for staging data sets:

- ▶ Uses the STG\_SIZE of the log stream specified on the LIKE parameter, if specified.
- ▶ Uses the maximum CF structure size for the structure to which the log stream is defined. This value is obtained from the value defined on the SIZE parameter for the structure in the CFRM policy.

Note that if both the STG\_DATACLAS and STG\_SIZE are specified, the value for STG\_SIZE overrides the space allocation attributes for the data class specified on the STG\_DATACLAS value.

Of all the staging data set-related parameters, STG\_SIZE is the most important. It is important to tune the size of the staging data sets when you first set up the log stream, and then re-check them every time you change the size of the associated CF structure or the number of log streams in that structure.

**Note:** Poor sizing of the DASD staging data sets can result in poor System Logger performance and inefficient use of resources. For more information about using the STG\_SIZE parameter, see 8.2, “Estimating log stream sizes” on page 274.

### **LS\_DATACLAS**

Specifies the name of the SMS data class that will be used when allocating the DASD offload data sets for this log stream.

If you specify LS\_DATACLAS(NO\_LS\_DATACLAS), which is the default, the data class is defined through SMS ACS routine processing. An SMS value specified on the LS\_DATACLAS parameter, including NO\_LS\_DATACLAS, always overrides one specified on a model log stream used on the LIKE parameter.

Whether it is preferable to have the DATACLAS association in the LOGR policy, or in the SMS ACS routines depends on your installation software management rules.

Either way, you need to provide a DATACLAS where SHAREOPTIONS (3,3) has been specified for the allocation of the offload data sets. SHAREOPTIONS (3,3) is required to allow you to fully share the data sets across multiple systems within the GRS complex. If your system is running in a monoplex configuration, SHAREOPTION (1,3), which is the default on the dynamic allocation call, is allowed.

See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS.

**Recommendation:** To ensure optimal I/O performance, we recommend you use a CISIZE of 24576 bytes for the offload data sets. The data class you specify on the LS\_DATACLAS parameter should be defined to use this CISIZE.

### LS\_MGMTCLAS

Specifies the name of the SMS management class to be used when allocating the offload data sets.

If you specify LS\_MGMTCLAS(NO\_LS\_MGMTCLAS), which is the default, the management class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the LS\_MGMTCLAS parameter, including NO\_LS\_MGMTCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS management classes, see Chapter 5 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### LS\_STORCLAS

Specifies the name of the SMS storage class to be used when allocating the offload data sets.

If you specify LS\_STORCLAS(NO\_LS\_STORCLAS), which is the default, the storage class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the LS\_STORCLAS parameter, including NO\_LS\_STORCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS storage classes, see Chapter 6 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### LS\_SIZE

Specifies the size, in 4 KB blocks, of the log stream offload DASD data sets for the log stream being defined.

When specified, this value will be used to specify a space allocation quantity on the log stream offload data set allocation request. It will override any size characteristics specified in an explicitly specified data class (via LS\_DATACLAS) or a data class that gets assigned via a DFSMS ACS routine.

The smallest valid LS\_SIZE value is 16 (64 KB), however in practice you would be unlikely to want such a small offload data set.

The largest size that a log stream offload data set can be defined for System Logger use is slightly less than 2 GB. If the data set size is too large, System Logger will automatically attempt to reallocate it smaller than 2 GB. The largest valid LS\_SIZE value is 524287.

If you omit LS\_SIZE, System Logger does one of the following, in the order listed, to allocate space for offload data sets:

- ▶ Uses the LS\_SIZE of the log stream specified on the LIKE parameter, if specified.
- ▶ Uses the size defined in the SMS data class for the log stream offload data sets.
- ▶ Uses dynamic allocation rules as specified in the ALLOCxx member of SYS1.PARMLIB for allocating data sets, if SMS is not available. Unless you specify otherwise in ALLOCxx, the default allocation amount is two tracks—*far* too small for most System Logger applications.

We recommend you explicitly specify an LS\_SIZE for each log stream even if you are using an SMS data class, as the value specified in the data class is unlikely to be appropriate for every log stream.

**Note:** Poor sizing of the offload data sets can result in System Logger performance issues, particularly during offload processing. For more information about using the LS\_SIZE parameter, see 8.2.1, “Sizing offload data sets” on page 274.

### AUTODELETE

Specifies when System Logger can physically delete log data.

If you specify AUTODELETE(NO), which is the default, System Logger physically deletes an offload data set only when *both* of the following are true:

- ▶ All the log data in the data set has been marked for deletion by a System Logger application using the IXGDELET service or by an archiving procedure like CICS/VR for CICS log streams or IFBEREPS, which is shipped in SYS1.SAMPLIB, for LOGREC log streams.
- ▶ The retention period for all the data in the offload data set has expired.

If you specify AUTODELETE(YES), System Logger automatically deletes log data whenever data is either marked for deletion (using the IXGDELET service) *or* the retention period for *all* the log data in the offload data set has expired.

Be careful when using AUTODELETE(YES) if the System Logger application manages log data deletion using the IXGDELET service. With AUTODELETE(YES), System Logger may delete data that the application expects to be accessible. If you specify AUTODELETE=YES with RETPD=0, data is eligible for deletion as soon as it is written to the log stream.

For more information about AUTODELETE and deleting log data, see 2.7, “Deleting log data” on page 64.

**Important:** Consult your System Logger application recommendations before using either the AUTODELETE or RETPD parameters. Setting these parameters inappropriately can result in significant performance and usability issues.

### RETPD

Specifies the retention period, in days, for log data in the log stream. The retention period begins when data is written to the *log stream*, not the offload data set. Once the retention period for all the log blocks in an offload data set has expired, the data set is eligible for physical deletion. The point at which System Logger physically deletes the data set depends on what you have specified on the AUTODELETE parameter and RETPD parameters:

- ▶ RETPD=0 - System Logger physically deletes expired offload data sets whenever offload processing is invoked, even if no log data is actually moved to any offload data sets.
- ▶ RETPD>0 - System Logger physically deletes expired offload data sets *only* when an offload data set fills and a new one gets allocated for the log stream.

System Logger will not process a retention period or delete data for log streams that are not connected and being written to by an application.

For example, a RETPD=1 would specify that any data written today would be eligible for deletion tomorrow. RETPD=10 would indicate data written today is eligible for deletion in 10 days.



The value specified for RETPD must be between 0 and 65,536.

For more information about RETPD and deleting log data, see 2.7, “Deleting log data” on page 64.

### **HLQ**

Specifies the high level qualifier for both the offload and staging data sets.

If you do not specify a high level qualifier, or if you specify HLQ(NO\_HLQ), the log stream will have a high level qualifier of IXGLOGR (the default value). If you specified the LIKE parameter, the log stream will have the high level qualifier of the log stream specified on the LIKE parameter. The value specified for HLQ overrides the high level qualifier for the log stream specified on the LIKE parameter.

HLQ and EHLQ are mutually exclusive and cannot be specified for the same log stream definition.

### **EHLQ**

Specifies the extended high level qualifier for both the offload and staging data sets. The EHLQ parameter was introduced by z/OS 1.4, and log streams specifying this parameter can only be connected to by systems running this level of z/OS or later.

EHLQ and HLQ are mutually exclusive and cannot be specified for the same log stream definition.

When the EHLQ parameter is not explicitly specified on the request, the resulting high level qualifier to be used for the log stream data sets will be based on whether the HLQ or LIKE parameters are specified. If the HLQ parameter is specified, then that value will be used for the offload data sets. When no high level qualifier is explicitly specified on the DEFINE LOGSTREAM request, but the LIKE parameter is specified, then the high level qualifier value being used in the referenced log stream will be used for the newly defined log stream. If the EHLQ, HLQ, and LIKE parameters are not specified, then the default value “IXGLOGR” will be used.

When EHLQ=NO\_EHLQ is specified or defaulted to, the resulting high level qualifier will be determined by the HLQ value from the LIKE log stream or using a default value.

See Example 2-1 for usage examples.

**CDS requirement:** The active primary TYPE=LOGR CDS must be formatted at a HBB705 or higher level in order to specify the EHLQ keyword. Otherwise, the request will fail. See “LOGR CDS format levels” on page 14 for more information.

#### *Example 2-1 Usage of EHLQ keyword*

1) Assume the OPERLOG log stream (NAME=SYSPLEX.OPERLOG) is defined with “EHLQ=MY.OWN.PREFIX”. The offload data set would be allocated as:

MY.OWN.PREFIX.SYSPLEX.OPERLOG.suffix

where the suffix is provided by System Logger.

2) Assume the OPERLOG log stream (NAME=SYSPLEX.OPERLOG) is attempted to be defined with “EHLQ=MY.PREFIX.IS.TOO.LONG”. Even though the EHLQ value is less than the maximum 33 characters, an offload data set cannot be allocated with an overall name greater than 44 characters. In this example, the define request would fail:

MY.PREFIX.IS.TOO.LONG.SYSplex.OPERLOG.suffix

The above data set name is not valid because it is too long.

---

### **HIGHOFFLOAD**

Specifies what percentage of the elements in the CF portion of the log stream can be used before offload processing is invoked. When the threshold is reached, System Logger begins offloading data from the CF structure to the offload data sets.

The default HIGHOFFLOAD value is 80%. If you omit the HIGHOFFLOAD parameter or specify HIGHOFFLOAD(0) the log stream will be defined with the default value.

The HIGHOFFLOAD value specified must be greater than the LOWOFFLOAD value.

For more information about offload thresholds and the offload process, see 2.6, “Offload processing” on page 58.

**Note:** For the recommended HIGHOFFLOAD values for various System Logger applications, refer to the chapter in this book that describes the subsystem you are interested in or consult the application’s manuals.

### **LOWOFFLOAD**

Specifies the point at which System Logger stops offloading CF log data to offload data sets. When offload processing has offloaded enough data that only this percentage of the elements in the CF portion of the log stream are being used, offload processing will end.

If you specify LOWOFFLOAD(0), which is the default, or omit the LOWOFFLOAD parameter, System Logger uses the 0% usage mark as the low offload threshold.

The value specified for LOWOFFLOAD must be less than the HIGHOFFLOAD value.

**Note:** For the recommended LOWOFFLOAD values for various System Logger applications, refer to the chapter in this book that describes the subsystem you are interested in or consult the application’s manuals.

### **MODEL**

Specifies whether the log stream being defined is a model, exclusively for use with the LIKE parameter to set up general characteristics for other log stream definitions.

If you specify MODEL(NO), which is the default, the log stream being defined is not a model log stream. Systems can connect to and use this log stream. The log stream can also be specified on the LIKE parameter, but is not exclusively for use as a model.

If you specify MODEL(YES), the log stream being defined is only a model log stream. It can be specified only as a model for other log stream definitions on the LIKE parameter.

Programs cannot connect to a log stream name that is defined as a model (MODEL(YES)) using an IXGCONN request.

No offload data sets are allocated for a model log stream.

The attributes of a model log stream are syntax checked at the time of the request, but not verified until another log stream references the model log stream on the LIKE parameter.

Model log streams can be thought of as a template for future log stream definitions. They are defined in the System Logger policy and will show up in output from a “D LOGGER,L” command or an IXCMIAPU LIST LOGSTREAM report. Some applications (such as CICS) use model log streams to allow installations to set up a log stream definition containing common characteristics that will be used as a basis for additional log streams that will be defined dynamically by the application (using the IXGINVNT service).

See Example 2-2 for an example of how a model log stream is used.

### LIKE

Specifies the name of another log stream defined in the System Logger policy. The characteristics of this log stream (such as storage class, management class, high level qualifier and so on) will be copied for the log stream you are defining if those characteristics are not explicitly coded on the referencing log stream. The parameters explicitly coded on this request, however, override the characteristics of the MODEL log stream specified on the *LIKE* parameter.

See Example 2-2 for an example of how a model log stream is used.

#### *Example 2-2 MODEL and LIKE usage*

---

The following statements will define a model log stream:

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(CFSTRM1.MODEL) MODEL(YES)
    STG_DUPLEX(NO) LS_SIZE(1000) HLQ(LOGHLQ) DIAG(YES)
    HIGHOFFLOAD(70) LOWOFFLOAD(10)
    AUTODELETE(YES)
```

with these attributes:

```
LOGSTREAM NAME(CFSTRM1.MODEL) STRUCTNAME() LS_DATACLAS()
    LS_MGMTCLAS() LS_STORCLAS() HLQ(LOGHLQ) MODEL(YES) LS_SIZE(1000)
    STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
    LOWOFFLOAD(10) HIGHOFFLOAD(70) STG_DUPLEX(NO) DUPLEXMODE()
    RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO) OFFLOADRECALL(YES)
    DASDONLY(NO) DIAG(YES) LOGGERDUPLEX(UNCOND) EHLQ(NO_EHLQ)
```

Here is a LIKE log stream definition based on the above model:

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(CFSTRM1.PROD) LIKE(CFSTRM1.MODEL)
    STRUCTNAME(LOG_TEST_001)
    STG_DUPLEX(YES) DUPLEXMODE(UNCOND) LOGGERDUPLEX(COND)
    EHLQ(LOG.EHLQ) OFFLOADRECALL(NO)
    LOWOFFLOAD(20)
```

that is defined with the following attributes:

```
LOGSTREAM NAME(CFSTRM1.PROD) STRUCTNAME(LOG_TEST_001) LS_DATACLAS()
    LS_MGMTCLAS() LS_STORCLAS() HLQ(NO_HLQ) MODEL(NO) LS_SIZE(1000)
    STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
    LOWOFFLOAD(20) HIGHOFFLOAD(70) STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
    RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO) OFFLOADRECALL(NO)
    DASDONLY(NO) DIAG(YES) LOGGERDUPLEX(COND) EHLQ(LOG.EHLQ)
```

Notice that while we inherited values from the MODEL log stream (HIGHOFFLOAD, LS\_SIZE, etc.), the values specified in the LIKE log stream definition will override them (EHLQ, STG\_DUPLEX, LOWOFFLOAD, etc.).

---

## **DIAG**

Specifies whether or not dumps or additional diagnostics should be provided by System Logger for certain conditions.

If you specify DIAG(NO), which is the default, this indicates that no special System Logger diagnostic activity is requested for this log stream, regardless of the DIAG specifications on the IXGCONN, IXGDELET and IXGBRWSE requests.

If you specify DIAG(YES), this indicates that special System Logger diagnostic activity is allowed for this log stream and can be obtained when the appropriate specifications are provided on the IXGCONN, IXGDELET, or IXGBRWSE requests.

We recommend that you specify DIAG(YES) as the additional diagnostics it provides will provide IBM service with additional information to debug problems. Note that specifying it will cause a slight performance impact when certain problems occur for which System Logger will now collect additional information; however, for the rest of the time, specifying DIAG(YES) has no effect on performance.

## **OFFLOADRECALL**

Indicates whether offload processing is to recall the current offload data set if it has been migrated.

If you specify OFFLOADRECALL(YES), offload processing will attempt to recall the current offload data set.

If you specify OFFLOADRECALL(NO), offload processing will skip recalling the current offload data set and allocate a new one.

If you use this option, you need to also consider how quickly your staging data sets get migrated, and how likely they are to be recalled. If they are likely to be migrated after just one offload, you should attempt to set LS\_SIZE to be roughly equivalent to the amount of data that gets moved in one offload. There is no point allocating a 1 GB offload data set, and then migrating it after you only write 50 MB of data into it. Similarly, if the data set is likely to be recalled (for an IXGBRWSE, for example), there is no point having a 1 GB data set clogging up your DASD if it only contains 50 MB of data. If the offload data sets will only typically contain 50 MB of data, then size them to be slightly larger than that.

How your log stream is used is a factor when deciding what value to code for OFFLOADRECALL. In general, if you can't afford the performance hit of having to wait for a offload data set to be recalled, we suggest you code OFFLOADRECALL(NO).

### ***Duplexing log data for CF-Structure based log streams***

Because the availability of log data is critical to many applications to allow successful recovery from failures, System Logger provides considerable functionality to ensure that log data managed by it will always be available if needed. When an application writes data to the log stream, it is duplexed while in interim storage until the data is "hardened" by offloading to the offload data sets. Duplexing ensures that the data is still available even if the CF containing the log stream were to fail.

For CF-Structure based log streams, System Logger *always* writes log data to the CF structure first. It will then duplex the data to some other medium (either another CF structure, staging data sets, or local buffers) depending on the values of several parameters in your log stream's definition as shown in Figure 2-9 on page 43. The following cases correspond to those in Figure 2-9 on page 43.

System-Managed Duplexing is *not* being used:

- ▶ Case 1: There is a failure-dependent connection between the connecting system and the structure, or, the CF containing the structure is volatile.
- ▶ Case 2: There is a failure-independent connection between the connecting system and the structure, *and*, the CF containing the structure is non-volatile.

**Note:** For a description of a failure dependent connection, refer to 2.8, “System Logger recovery” on page 67.

System-Managed Duplexing is being used:

There are two instances of the structure.

- ▶ Case 3: There is a failure-dependent connection between the connecting system and the composite structure view<sup>1</sup>, or, the structure CF state is volatile.
- ▶ Case 4: There is a failure-independent connection between the connecting system and the composite structure view, and a failure dependent relationship between the two structure instances, and the structure CF state is non-volatile.
- ▶ Case 5: There is a failure-independent connection between the connecting system and the composite structure view, and a failure independent relationship between the two structure instances, and the structure CF state is non-volatile.

<u>Location of duplex copy of log data</u>				
Mode / Case #	LoggerDuplex	Stg_Duplex (No)	Stg_Duplex(Yes) DuplexMode (Cond)	Stg_Duplex(Yes) DuplexMode (Uncond)
		<i>simplex</i>		
Case 1	Uncond   Cond	Local Buffers	Staging Data Set	Staging Data Set
Case 2	Uncond   Cond	Local Buffers	Local Buffers	Staging Data Set
<i>duplex</i>				
Case 3	Uncond   Cond	Structure, Local Buffers	Structure, Staging Data Set	Structure, Staging Data Set
Case 4	Uncond   Cond	Structure, Local Buffers	Structure, Local Buffers	Structure, Staging Data Set
Case 5a	Uncond	Structure, Local Buffers	Structure, Local Buffers	Structure, Staging Data Set
Case 5b	Cond	Structure	Structure	Structure

Figure 2-9 System Logger duplexing combinations

<sup>1</sup> The composite structure view means that XES first determines the relationship between two System-Managed Duplexed structures and provides the composite view to the connector:

structure instance 1	structure instance 2 (duplexed copy of 1)	composite view to connecting system
failure independent	failure independent	failure independent
failure independent	failure dependent	failure independent
failure dependent	failure independent	failure independent
failure dependent	failure dependent	failure dependent

It is possible that System Logger could start out duplexing to one medium (say, local buffers) and some event (CF becomes volatile, losing a CF, update of the above parameters in log stream definition, and so on) could cause the transition from one duplexing medium to another. For example, a log stream defined with STG\_DUPLEX(YES), DUPLEXMODE(COND) and LOGGERDUPLEX(COND) to a structure falling under the category of case 5 could lose connectivity to one of the CFs and fall back to a simplex mode case 2 (assuming the remaining CF/system view is failure independent) and System Logger would begin duplexing to its local buffers. For more information about how CF state changes can effect System Logger duplexing, see “Structure state changes” on page 72.

You can view the storage mediums System Logger is duplexing to by entering the “D LOGGER,C,LSN=logstreamname,DETAIL” system command. See Example 2-3 for sample output. Note that the locations listed on the DUPLEXING: line are where the *copy* of the data is held. For example, in Example 2-3, the OPERLOG structure is being duplexed using System-Managed Duplexing, however the CFs are not failure isolated from System Logger, so System Logger actually has three copies of the data in this case—one in the CF structure (not shown in the command), a second in the duplex copy of that structure (shown by the keyword STRUCTURE), and a third copy in the staging data set (shown by the keyword STAGING DATA SET).

*Example 2-3 Example output of display command showing duplex mediums*

---

```
IXG601I 14.20.30  LOGGER DISPLAY 555
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #0$3
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----
SYSPLEX.OPERLOG    LOG_TEST_001      000002  IN USE
  DUPLEXING: STRUCTURE, STAGING DATA SET
  JOBNAME: SICA     ASID: 0082
  R/W CONN: 000001 / 000000
  RES MGR./CONNECTED: *NONE* / NO
  IMPORT CONNECT: NO
  JOBNAME: CONSOLE  ASID: 000A
  R/W CONN: 000000 / 000001
  RES MGR./CONNECTED: *NONE* / NO
  IMPORT CONNECT: NO
```

---

## 2.5.2 DASD-only log streams

DASD-only log streams are single system in scope; that is, only one system in the sysplex can be connected at a given time, although multiple applications from the *same* system can be connected simultaneously. These log streams use local buffers in System Logger's data space for interim storage. Data is then offloaded to offload data sets for longer term storage. Figure 2-10 shows how a DASD-only log stream spans the two levels of storage.

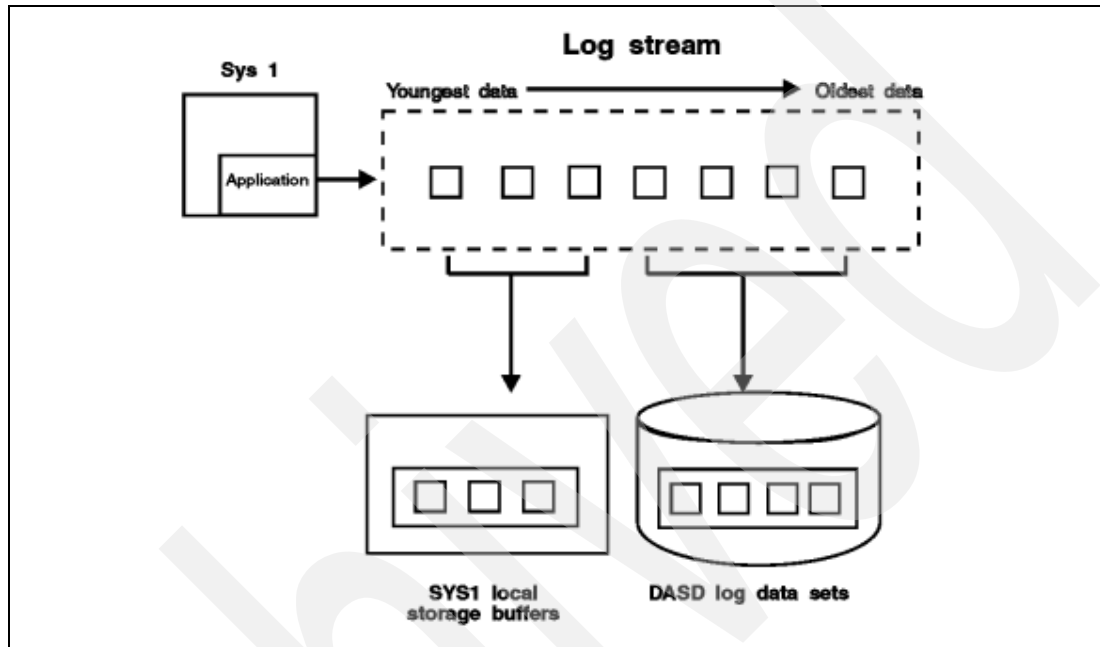


Figure 2-10 DASD-only log stream spanning two levels of storage

When an application writes a log block to a DASD-only log stream, System Logger first writes the data to local buffers; the log block is then *always* duplexed to staging data sets. When the staging data set subsequently reaches the installation-defined high threshold, System Logger moves the log blocks from local buffers to offload data sets and deletes those log blocks from the staging data set. From an application point of view, the actual location of the log data in the log stream is transparent.

There are many considerations you must take into account before and after you have decided to use DASD-only log streams. If you decide your System Logger application should use them, then you should also have a basic understanding of how they work. We'll spend the rest of this section addressing these points.

If you have determined that all your log streams should use CF-Structure log streams, you should skip to 2.5.3, "Updating log stream definitions" on page 55 now.

### When your application should use DASD-only log streams

Earlier we discussed some factors to be taken into consideration when deciding what type of log stream your application should use:

- ▶ The location and concurrent activity of writers and readers to a log stream's log data.
- ▶ The volume (data flow) of log stream data.

In addition to these, you should consider any advice given by the application; for example, applications that require that multiple systems connect to the log stream, like APPC/MVS for

example, may require that the log stream is kept in a CF. In general though, DASD-only log streams can be used when:

- ▶ There is no requirement to have more than one system in the sysplex accessing the log stream at the same time.
- ▶ There are lower volumes of log data being written to the log stream.

**Note:** A DASD-only log stream is single system in scope. This means that even though there can be multiple connections to it from a *single* system in the sysplex, there cannot be multiple *systems* connected to the log stream at the same time.

## Implementing DASD-only log streams

There are many parameters to consider when defining a log stream. We cover the specifics of each in this section; note that some of this material repeats information from the CF-Structure based log stream section; any differences are indicated in Table 2-3 on page 22.

### Defining DASD-only log streams

DASD-only log streams are also defined using the IXCMIAPU utility or the IXGINVNT service. Let us take a look at the different parameters and how they impact System Logger operations.

#### NAME

Specifies the name of the log stream that you want to define. The name can be made up of one or more segments separated by periods, up to the maximum length of 26 characters.

NAME is a required parameter with no default. Besides being the name of the log stream, it is used as part of the offload data set and staging data set names:

*offload data set example:* <hlq>.logstreamname.<seq#>  
*staging data set example:* <hlq>.logstreamname.<sysplex\_name>

Note that if the log stream name combined with the HLQ are longer than 33 characters, the data component of the offload and maybe the staging data sets may contain a system-generated low level qualifier: for example:

IXGLOGR.A2345678.B2345678.C2345678.A0000000 (Cluster)  
IXGLOGR.A2345678.B2345678.C2345678.IE8I36RM (Data)

#### RMNAME

Specifies the name of the resource manager program associated with the log stream. *RNAME* must be 8 alphanumeric or national (\$, #, or @) characters, padded on the right with blanks if necessary. You must define RMNAME in the System Logger policy before the resource manager can connect to the log stream. See the System Logger chapter in *z/OS MVS Assembler Services Guide*, SA22-7605, for information about writing a resource manager program to back up a log stream.

#### DESCRIPTION

Specifies user-defined data describing the log stream. *DESCRIPTION* must be 16 alphanumeric or national (\$, #, @) characters, underscore (\_) or period (.), padded on the right with blanks if necessary.

#### DASDONLY

Specifies whether the log stream being defined is a CF or a DASD-only log stream. This is an optional parameter with the default being DASDONLY(NO).

Since we are reviewing DASD-only log streams in this section, DASDONLY(YES) would be used to indicate that we do want a DASD-only log stream.



### **MAXBUFSIZE**

Specifies the size, in bytes, of the largest log block that can be written to the DASD-only log stream being defined in this request.

The value for MAXBUFSIZE must be between 1 and 65,532 bytes. The default is 65,532 bytes.

This parameter is only valid with DASDONLY(YES).

There are some additional considerations for MAXBUFSIZE if you plan on migrating the log stream to a CF structure at some point. See “Migrating a log stream from DASD-only to CF-Structure based” on page 56.

**Important:** Remember, STG\_xx parameters only apply to staging data sets; *not* to offload data sets. LS\_xx parameters are used to associate SMS constructs with offload data sets.

### **STG\_DATACLAS**

Specifies the name of the SMS data class that will be used when allocating the DASD staging data sets for this log stream.

If you specify STG\_DATACLAS(NO\_STG\_DATACLAS), which is the default, the data class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the STG\_DATACLAS parameter, including NO\_STG\_DATACLAS, always overrides one specified on a model log stream used on the LIKE parameter.

Remember to use SHAREOPTIONS(3,3) when defining your SMS data class!

For information about defining SMS data classes, see Chapter 7 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

Remember to use SHAREOPTIONS(3,3) when defining your SMS data class!

For information about defining SMS data classes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402, Chapter 7.

**Important:** Do not change the Control Interval Size attributes for staging data sets to be anything other than 4096. System Logger requires that the Control Interval Size for staging data sets be set to 4096. If staging data sets are defined with characteristics that result in a Control Interval Size other than 4096, System Logger will not use those data sets to keep a duplicate copy of log data. Operations involving staging data sets defined with a Control Interval Size other than 4096 will fail to complete successfully. The DASD-only log stream will be unusable until this is corrected.

### **STG\_MGMTCLAS**

Specifies the name of the SMS management class used when allocating the staging data set for this log stream.

If you specify STG\_MGMTCLAS(NO\_STG\_MGMTCLAS), which is the default, the management class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the STG\_MGMTCLAS parameter, including NO\_STG\_MGMTCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS management classes, see Chapter 5 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### **STG\_STORCLAS**

Specifies the name of the SMS storage class used when allocating the DASD staging data set for this log stream.

If you specify `STG_STORCLAS(NO_STG_STORCLAS)`, which is the default, the storage class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410, for more information about SMS. An SMS value specified on the `STG_STORCLAS` parameter, including `NO_STG_STORCLAS`, always overrides one specified on a model log stream used on the `LIKE` parameter.

For information about defining SMS storage classes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402, Chapter 6.

### **STG\_SIZE**

Specifies the size, in 4 KB blocks, of the DASD staging data set for the log stream being defined.

When specified, this value will be used to specify a space allocation quantity on the log stream staging data set allocation request. It will override any size characteristics specified in the specified data class (via `STG_DATACLAS`) or a data class that gets assigned via a DFSMS ACS routine.

Note that if both the `STG_DATACLAS` and `STG_SIZE` are specified, the value for `STG_SIZE` overrides the space allocation attributes for the data class specified on the `STG_DATACLAS` value.

If you omit `STG_SIZE` for a DASD-only log stream, System Logger does one of the following, in the order listed, to allocate space for staging data sets:

- ▶ Uses the `STG_SIZE` of the log stream specified on the `LIKE` parameter, if specified.
- ▶ Uses the size defined in the SMS data class for the staging data sets.
- ▶ Uses dynamic allocation rules (as defined in the `ALLOCxx` member of `Parmlib`) for allocating data sets if the SMS ACS routines do not assign a data class.

**Note:** Sizing DASD staging data sets incorrectly can cause System Logger performance issues. For more information about using the `STG_SIZE` parameter, see 8.2.2, “Sizing interim storage” on page 275.

### **LS\_DATACLAS**

Specifies the name of the SMS data class that will be used when allocating offload data sets.

If you specify `LS_DATACLAS(NO_LS_DATACLAS)`, which is the default, the data class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the `LS_DATACLAS` parameter, including `NO_LS_DATACLAS`, always overrides one specified on a model log stream used on the `LIKE` parameter.

System Logger uses VSAM linear data sets for offload data sets. They require a control interval (CISIZE) from 4096 to 32768 bytes, and it can be expressed in increments of 4096; the default is 4096.

Remember to specify `SHAREOPTIONS(3,3)` when defining your SMS data class.

For information about defining SMS data classes, see Chapter 7 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

**Recommendation:** If you want to ensure optimal I/O performance, we recommend you use a CISIZE of 24576 bytes. You can specify a DFSMS data class that is defined with a control interval size of 24576 on the LS\_DATACLAS parameter of a log stream definition to have offload data sets allocated with control interval sizes of 24576.

### **LS\_MGMTCLAS**

Specifies the name of the SMS management class to be used when allocating the offload data sets.

If you specify LS\_MGMTCLAS(NO\_LS\_MGMTCLAS), which is the default, the management class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410 for more information about SMS. An SMS value specified on the LS\_MGMTCLAS parameter, including NO\_LS\_MGMTCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS management classes, see Chapter 5 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### **LS\_STORCLAS**

Specifies the name of the SMS storage class to be used when allocating the offload data sets.

If you specify LS\_STORCLAS(NO\_LS\_STORCLAS), which is the default, the storage class is defined by standard SMS processing. See *z/OS DFSMS: Using Data Sets*, SC26-7410, for more information about SMS. An SMS value specified on the LS\_STORCLAS parameter, including NO\_LS\_STORCLAS, always overrides one specified on a model log stream used on the LIKE parameter.

For information about defining SMS storage classes, see Chapter 6 in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### **LS\_SIZE**

Specifies the size, in 4 KB blocks, of the log stream offload DASD data sets for the log stream being defined.

When specified, this value will be used to specify a space allocation quantity on the log stream offload data set allocation request. It will override any size characteristics specified in an explicitly specified data class (via LS\_DATACLAS) or a data class that gets assigned via a DFSMS ACS routine.

The smallest valid LS\_SIZE value is 16 (64 KB). However, in practice you would be unlikely to want such a small offload data set.

The largest size that a log stream offload data set can be defined for System Logger use is slightly less than 2 GB. If the data set size is too large, System Logger will automatically attempt to reallocate it smaller than 2 GB. The largest valid LS\_SIZE value is 524287.

If you omit LS\_SIZE, System Logger does one of the following, in the order listed, to allocate space for offload data sets:

- ▶ Uses the LS\_SIZE of the log stream specified on the LIKE parameter, if specified.
- ▶ Uses the size defined in the SMS data class for the log stream offload data sets.

- ▶ Uses dynamic allocation rules as specified in the ALLOCxx member of SYS1.PARMLIB for allocating data sets, if SMS does not assign a data class. Unless you specify otherwise in ALLOCxx, the default allocation amount is two tracks—*far* too small for most System Logger applications.

We recommend you explicitly specify an LS\_SIZE for each log stream even if you are using an SMS data class, as the value specified in the data class may not be suited to your log stream.

**Note:** Sizing offload data sets incorrectly can cause System Logger performance issues. For more information about using the LS\_SIZE parameter, see 8.2.1, “Sizing offload data sets” on page 274.

### AUTODELETE

Specifies when System Logger can physically delete log data.

If you specify AUTODELETE(NO), which is the default, System Logger physically deletes an offload data set only when *both* of the following are true:

- ▶ All the log data in the data set has been marked for deletion by a System Logger application using the IXGDELET service or by an archiving procedure like CICS/VR for CICS log streams or IFBEREPS, which is shipped in SYS1.SAMPLIB, for LOGREC log streams.
- ▶ The retention period for all the data in the offload data set has expired.

If you specify AUTODELETE(YES), System Logger automatically deletes log data whenever data is either marked for deletion (using the IXGDELET service) or the retention period for all the log data in a data set has expired.

Be careful when using AUTODELETE(YES) if the System Logger application manages log data deletion using the IXGDELET service. With AUTODELETE(YES), System Logger may delete data that the application expects to be accessible. If you specify AUTODELETE=YES with RETPD=0, data is eligible for deletion as soon as it is written to the log stream.

For more information about AUTODELETE and deleting log data, see 2.7, “Deleting log data” on page 64.

**Important:** Consult your System Logger application recommendations before using either the AUTODELETE or RETPD parameters.

### RETPD

Specifies the retention period, in days, for log data in the log stream. The retention period begins when data is written to the *log stream*, not the offload data set. Once the retention period for an entire offload data set has expired, the data set is eligible for physical deletion. The point at which System Logger physically deletes the data set depends on what you have specified on the AUTODELETE parameter and RETPD parameters:

- ▶ RETPD=0 - System Logger physically deletes expired offload data sets whenever offload processing is invoked, even if no log data is actually moved to any offload data sets.
- ▶ RETPD>0 - System Logger physically deletes expired offload data sets *only* when an offload data set fills and it switches to a new one for a log stream.

System Logger will not process a retention period or delete data for log streams that are not connected and being written to by an application.

For example, a RETPD=1 would specify that any data written today would be eligible for deletion tomorrow. RETPD=10 would indicate data written today is eligible for deletion in 10 days.

The value specified for RETPD must be between 0 and 65,536.

For more information about RETPD and deleting log data, see 2.7, “Deleting log data” on page 64.

### **HLQ**

Specifies the high level qualifier for both the offload and staging data sets.

If you do not specify a high level qualifier, or if you specify HLQ(NO\_HLQ), the log stream will have a high level qualifier of IXGLOGR (the default value). If you specified the LIKE parameter, the log stream will have the high level qualifier of the log stream specified on the LIKE parameter. The value specified for HLQ overrides the high level qualifier for the log stream specified on the LIKE parameter.

HLQ and EHLQ are mutually exclusive and cannot be specified for the same log stream definition.

### **EHLQ**

Specifies the extended high level qualifier for both the offload and staging data sets. The *EHLQ* parameter was introduced by z/OS 1.4, and log streams specifying this parameter can only be connected to by systems running this level of z/OS or later.

EHLQ and HLQ are mutually exclusive and cannot be specified for the same log stream definition.

When the EHLQ parameter is not explicitly specified on the request, the resulting high level qualifier to be used for the log stream data sets will be based on whether the HLQ or LIKE parameters are specified. If the HLQ parameter is specified, then that value will be used for the offload data sets. When no high level qualifier is explicitly specified on the DEFINE LOGSTREAM request, but the LIKE parameter is specified, then the high level qualifier value being used in the referenced log stream will be used for the newly defined log stream. If the EHLQ, HLQ, and LIKE parameters are not specified, then the default value “IXGLOGR” will be used.

When EHLQ=NO\_EHLQ is specified or defaulted to, the resulting high level qualifier will be determined by the HLQ value from the LIKE log stream or using a default value.

See Example 2-1 on page 39 for usage examples.

**CDS requirement:** The active primary TYPE=LOGR CDS must be formatted at a HBB705 or higher level in order to specify the EHLQ keyword. Otherwise, the request will fail. See “LOGR CDS format levels” on page 14 for more information.

#### *Example 2-4 Use of EHLQ*

1) Assume the OPERLOG log stream (NAME=SYSPLEX.OPERLOG) is defined with “EHLQ=MY.OWN.PREFIX” specified. The log stream data set would be allocated as:

```
MY.OWN.PREFIX.SYSPLEX.OPERLOG.suffix
```

where the suffix is up to an eight-character field provided by System Logger.

2) Assume the OPERLOG log stream (NAME=SYSPLEX.OPERLOG) is attempted to be defined with “EHLQ=MY.PREFIX.IS.TOO.LONG”. Even though the EHLQ value is less than the maximum 33 characters, an offload data set cannot be allocated with an overall name greater than 44 characters. In this example, the define request would fail:

```
MY.PREFIX.IS.TOO.LONG.SYSPLEX.OPERLOG.suffix
```

The above data set name is not valid because it is too long.

---

### **HIGHOFFLOAD**

Specifies the point at which offload processing should start for the log stream. This is specified in terms of percent full for the staging data set. For example, if HIGHOFFLOAD is set to 80, offload processing will start when the staging data set reaches 80% full.

The default HIGHOFFLOAD value is 80%. If you omit the HIGHOFFLOAD parameter or specify HIGHOFFLOAD(0) the log stream will be defined with the default value.

The HIGHOFFLOAD value specified must be greater than the LOWOFFLOAD value.

For more information about offload thresholds and the offload process, see 2.6, “Offload processing” on page 58.

**Note:** For System Logger application recommended HIGHOFFLOAD values, see the chapter in this book that describes the subsystem you are interested in, or consult the application’s manuals.

### **LOWOFFLOAD**

Specifies the point at which offload processing will end. This is specified in terms of percent full for the staging data set. For example, if LOWOFFLOAD is set to 10, offload processing will end when enough data has been moved so that the staging data set is now only 10% full.

If you specify LOWOFFLOAD(0), which is the default, or omit the LOWOFFLOAD parameter, System Logger continues offloading until the staging data set is empty.

The value specified for LOWOFFLOAD must be less than the HIGHOFFLOAD value.

For more information about offload thresholds and the offload process, see 2.6, “Offload processing” on page 58.

**Note:** For System Logger application recommended LOWOFFLOAD values, see the chapter in this book that describes the subsystem you are interested in or consult the application’s manuals.

### **MODEL**

Specifies whether the log stream being defined is a model, exclusively for use with the LIKE parameter to set up general characteristics for other log stream definitions.

If you specify MODEL(NO), which is the default, the log stream being defined is not a model log stream. Systems can connect to and use this log stream. The log stream can also be specified on the LIKE parameter, but is not exclusively for use as a model.

If you specify MODEL(YES), the log stream being defined is only a model log stream. It can be specified only as a model for other log stream definitions on the LIKE parameter.

Programs cannot connect to a log stream name that is defined as a model (MODEL(YES)) using an IXGCONN request.

No log stream offload data sets are allocated on behalf of a model log stream.

The attributes of a model log stream are syntax checked at the time of the request, but not verified until a another log stream references the model log stream on the LIKE parameter.

Model log streams can be thought of as a template for future log stream definitions. They are defined in the System Logger policy and will show up in output from a "D LOGGER,L" command or an IXCMIAPI LIST LOGSTREAM report. Some applications (such as CICS) use model log streams to allow installations to set a common group of characteristics up for the application to define LIKE log streams based on.

See Example 2-2 on page 41 for usage information.

### LIKE

Specifies the name of a log stream defined in the System Logger policy. The characteristics of this log stream (such as storage class, management class, high level qualifier and so on) will be copied for the log stream you are defining only if those characteristics are not explicitly coded on the referencing log stream. The parameters explicitly coded on this request, however, override the characteristics of the MODEL log stream specified on the LIKE parameter.

See Example 2-2 on page 41 for usage information.

#### *Example 2-5 MODEL and LIKE usage*

---

The following JCL will define a model log stream:

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(CFSTRM1.MODEL) MODEL(YES)
  STG_DUPLEX(NO) LS_SIZE(1000) HLQ(LOGHLQ) DIAG(YES)
  HIGHOFFLOAD(70) LOWOFFLOAD(10)
  AUTODELETE(YES)
```

with these attributes:

```
LOGSTREAM NAME(CFSTRM1.MODEL) STRUCTNAME() LS_DATACLAS()
  LS_MGMTCLAS() LS_STORCLAS() HLQ(LOGHLQ) MODEL(YES) LS_SIZE(1000)
  STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
  LOWOFFLOAD(10) HIGHOFFLOAD(70) STG_DUPLEX(NO) DUPLEXMODE()
  RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(YES) OFFLOADRECALL(YES)
  DASDONLY(NO) DIAG(YES) LOGGERDUPLEX(UNCOND) EHLQ(NO_EHLQ)
```

Here is a LIKE log stream definition based on the above model:

```
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM NAME(CFSTRM1.PROD) LIKE(CFSTRM1.MODEL)
  STRUCTNAME(LOG_TEST_001)
  STG_DUPLEX(YES) DUPLEXMODE(UNCOND) LOGGERDUPLEX(COND)
  EHLQ(LOG.EHLQ) OFFLOADRECALL(NO)
  LOWOFFLOAD(20) AUTODELETE(NO)
```

that is defined with the following attributes:

```
LOGSTREAM NAME(CFSTRM1.PROD) STRUCTNAME(LOG_TEST_001) LS_DATACLAS()
  LS_MGMTCLAS() LS_STORCLAS() HLQ(NO_HLQ) MODEL(NO) LS_SIZE(1000)
  STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS() STG_SIZE(0)
  LOWOFFLOAD(20) HIGHOFFLOAD(70) STG_DUPLEX(YES) DUPLEXMODE(UNCOND)
```

RMNAME() DESCRIPTION() RETPD(0) AUTODELETE(NO) OFFLOADRECALL(NO)  
DASDONLY(NO) DIAG(YES) LOGGERDUPLEX(COND) EHLQ(LOG.EHLQ)

Notice that while we inherited values from the MODEL log stream(HIGHOFFLOAD, LS\_SIZE, etc), the values specified in the LIKE log stream definition will override them (EHLQ, STG\_DUPLEX, LOWOFFLOAD, etc).

---

### **DIAG**

Specifies whether or not dumps or additional diagnostics should be provided by System Logger for certain conditions.

If you specify DIAG(NO), which is the default, this indicates that no special System Logger diagnostic activity is requested for this log stream, regardless of the DIAG specifications on the IXGCONN, IXGDELET and IXGBRWSE requests.

If you specify DIAG(YES), this indicates that special System Logger diagnostic activity is allowed for this log stream and can be obtained when the appropriate specifications are provided on the IXGCONN, IXGDELET, or IXGBRWSE requests.

We recommend that you specify DIAG(YES) as the additional diagnostics it provides will provide IBM service with additional information to debug problems. Note that specifying it will cause a slight performance impact when certain problems occur for which System Logger will now collect additional information; however, for the rest of the time, specifying DIAG(YES) has no effect on performance.

### **OFFLOADRECALL**

Indicates whether offload processing is to recall the current offload data set if it has been migrated.

If you specify OFFLOADRECALL(YES), offload processing will attempt to recall the current offload data set.

If you specify OFFLOADRECALL(NO), offload processing will skip recalling the current offload data set and allocate a new one.

If you use this option, you need to also consider how quickly your staging data sets get migrated, and how likely they are to be recalled. If they are likely to be migrated after just one offload, you should attempt to set LS\_SIZE to be roughly equivalent to the amount of data that gets moved in one offload. There is no point allocating a 1 GB offload data set, and then migrating it after you only write 50 MB of data into it. Similarly, if the data set is likely to be recalled (for an IXGBRWSE, for example), there is no point having a 1 GB data set clogging up your DASD if it only contains 50 MB of data. If the offload data sets will only typically contain 50 MB of data, then size them to be slightly larger than that.

How your log stream is used is a factor when deciding what value to code for OFFLOADRECALL. In general, if you can't afford the performance hit of having to wait for a offload data set to be recalled, we suggest you code OFFLOADRECALL(NO).

### ***Duplexing log data for DASD-only log streams***

When an application writes data to the log stream, it is duplexed while in interim storage until the data is offloaded. Duplexing prevents log data being lost because of the loss of a single point of failure.

For DASD-only log streams, System Logger uses local buffers in System Logger's data space for interim storage. It then duplexes the data simultaneously to staging data sets. Unlike CF-Structure based log streams, you have no control over this processing; System Logger always uses this configuration for DASD-only log streams.



## 2.5.3 Updating log stream definitions

Log stream definitions are updated using the IXCMIAPU utility or the IXGINVNT service. When and how you can update the log stream definition varies depending on the level of the LOGR CDS you are using.

If you are running with an HBB6603 or lower LOGR CDS, most log stream attributes cannot be updated if there is any type of log stream connection, whether it is active or “failed-persistent”. Use of the log stream in question needs to be quiesced before submitting the update requests. The exception to this is the RETPD and AUTODELETE parameters, which can be updated at any time, with the values taking effect when the next offload data set switch occurs (that is, when System Logger allocates a new offload data set). For CF-Structure based log streams, changing the CF structure a log stream resides in is a cumbersome process that involves deleting the log stream and redefining it to the new structure.

If you are running with an HBB7705 or higher LOGR CDS, System Logger allows updates to be submitted at *any* time for offload and connection-based log stream attributes. The updates become “pending updates” and are shown as such in the IXCMIAPU LIST LOGSTREAM report. The updates are then committed at different times, depending on which parameter is being changed. Log streams without any connections will have their updates go into effect immediately. Table 2-4 shows which parameters can be changed, and when the change takes effect.

Example 2-6 on page 56 contains a sample job that updates some log stream attributes.

Table 2-4 System Logger logstream attribute dynamic update commit outline

Logstream attribute	Last disconnect or first connect to logstream in sysplex	Switch to New offload Data Set	CF Structure Rebuild
RETPD	Yes	Yes	No
AUTODELETE	Yes	Yes	No
LS_SIZE	Yes	Yes	Yes
LS_DATACLAS	Yes	Yes	Yes
LS_MGMTCLAS	Yes	Yes	Yes
LS_STORCLAS	Yes	Yes	Yes
OFFLOADRECALL	Yes	Yes (1)	Yes
LOWOFFLOAD	Yes	Yes (1)	Yes
HIGHOFFLOAD	Yes	Yes (1)	Yes
STG_SIZE	Yes	No	Yes
STG_DATCLAS	Yes	No	Yes
STG_MGMTCLAS	Yes	No	Yes
STG_STORCLAS	Yes	No	Yes
STG_DUPLEX (CF)	Yes	No	No
DUPLEXMODE (CF)	Yes	No	No

Logstream attribute	Last disconnect or first connect to logstream in sysplex	Switch to New offload Data Set	CF Structure Rebuild
LOGGERDUPLEX (CF)	Yes	No	No
MAXBUFSIZE (DO)	Yes	No	N/a

Notes:  
1 - These attributes are only committed during switch to new offload data set activity for DASD-only log stream. These attributes are not committed at this point for CF structure-based log stream.

yes - Indicates the attribute is committed during the activity listed in the column heading.

no - Indicates the attribute is not committed during the activity

(CF) - Indicates the attribute is only applicable to CF structure-based log stream

(DO) - Indicates the attribute is only applicable to DASD-only-based log stream

*Example 2-6 Example update job and output*

Example UPDATE LOGSTREAM request:

```
DATA TYPE(LOGR) REPORT(YES)
UPDATE LOGSTREAM NAME(SYSPLEX.OPERLOG)
DUPLXMODE(UNCOND)
OFFLOADRECALL(YES)
```

Example IXCMIAPU report output showing pending updates:

```
LOGSTREAM NAME(SYSPLEX.OPERLOG) STRUCTNAME(LOG_TEST_001) LS_DATACLAS(LOGR24K)
LS_MGMTCLAS() LS_STORCLAS() HLQ(NO_HLQ) MODEL(NO) LS_SIZE(1024)
STG_MGMTCLAS() STG_STORCLAS() STG_DATACLAS(LOGR4K) STG_SIZE(0)
LOWOFFLOAD(0) HIGHOFFLOAD(80) STG_DUPLEX(YES) DUPLEXMODE(COND)
RMNAME() DESCRIPTION() RETPD(2) AUTODELETE(YES) OFFLOADRECALL(NO)
DASDONLY(NO) DIAG(NO) LOGGERDUPLEX(COND) EHLQ(IXGLOGR)

PENDING CHANGES:
OFFLOADRECALL(YES)
DUPLEXMODE(UNCOND)
```

This support (again, HBB7705 LOGR CDS required) also introduced the ability to dynamically change the CF structure the log stream was defined to. This can be done *without* first deleting the log stream. Note that the log stream cannot have any active or “failed persistent” connections for the update to be honored.

### **Migrating a log stream from DASD-only to CF-Structure based**

It is possible to migrate a DASD-only log stream to a CF structure based one without deleting and redefining it; the log stream then is a CF-Structure based log stream and will operate as such. The migration is done using the IXCMIAPU utility to update the DASD-only log stream definition to refer to a CF structure by specifying the STRUCTNAME parameter as shown in Example 2-7 on page 57.

**Note:** A DASD-only log stream automatically duplexes log data to DASD staging data sets. When you upgrade a DASD-only log stream to a CF-based log stream, you will still get duplexing to staging data sets unless you specify otherwise. You must specify STG\_DUPLEX(NO) when you upgrade on the UPDATE LOGSTREAM request to get a CF-Structure based log stream that duplexes to local buffers rather than to DASD staging data sets.

For the update to be honored, all connections (active and “failed persistent”) must be disconnected. Before updating the log stream definition, you should be familiar with all the concepts discussed in 2.5.1, “CF-Structure based log streams” on page 23. Also, the CF structure you intend to migrate the DASD-only log stream to must meet some requirements:

- ▶ When defining DASD-only log streams, plan ahead for possible future upgrades by matching the MAXBUFSIZE value for the DASD-only log stream with the MAXBUFSIZE value of the structure you would assign it to on an upgrade request. The MAXBUFSIZE value on the DASD-only log stream definition must be the same size as, or smaller than, the MAXBUFSIZE value for the structure.
- ▶ On the UPDATE request to upgrade a DASD-only log stream, specify a structure with a MAXBUFSIZE value that is as large as, or larger than, the DASD-only log stream MAXBUFSIZE value.

**Restriction:** You cannot issue an UPDATE request to reduce the MAXBUFSIZE value on a DASD-only log stream definition. You also cannot specify the MAXBUFSIZE parameter on an UPDATE request for a structure definition.

*Example 2-7 Migrating a DASD-only log stream*

Assume we have a DASD-only log stream defined called DOSTRM1.PROD We would then submit the following job to update the log stream to be defined in the CF structure called LOG\_TEST\_001 (assuming the request is valid; i.e. MAXBUFSIZE ok, no connections).

```
DATA TYPE(LOGR) REPORT(YES)
UPDATE LOGSTREAM NAME(DOSTRM1.PROD)
STRUCTNAME(LOG_TEST_001)
```

We can then use the D LOGGER,L command the verify the log stream has been migrated:

```
IXG601I 16.36.43  LOGGER DISPLAY 533
INVENTORY INFORMATION BY LOGSTREAM
```

LOGSTREAM	STRUCTURE	#CONN	STATUS
DOSTRM1.PROD	LOG_TEST_001	000000	AVAILABLE

**Note:** It is *not* possible to migrate from a CF-Structure based log stream to a DASD-only log stream without deleting and re-defining the log stream.

## 2.5.4 Deleting log stream and CF structure definitions

Both log streams and CF structures are deleted using either the IXCMIAPU utility or the IXGINVNT service. This section discusses how to use these tools to delete the System Logger construct, while 2.7, “Deleting log data” on page 64, details how log data within a log stream is disposed of.

## Deleting log streams

The DELETE LOGSTREAM command requests that an entry for a log stream (complete with all associated staging and offload data sets) be deleted from the System Logger policy. The process is the same for both DASD-only and CF-Structure based log streams.

There is only one parameter on the DELETE LOGSTREAM request:

### NAME

Specifies the name of the log stream you want to delete from the System Logger policy.

Example 2-8 shows a sample DELETE LOGSTREAM request.

*Example 2-8 Sample log stream delete statements*

---

```
DATA TYPE(LOGR) REPORT(YES)
DELETE LOGSTREAM NAME(CFSTRM2.PROD)
```

---

You cannot delete a log stream while there are any active connections to it. You also cannot delete a log stream that has a “failed-persistent” connection if System Logger is unable to resolve the connection. Remember that once you delete the log stream, all the data in any offload data sets associated with the log stream will also be gone. So, if you need that data, it is your responsibility to copy it from the log stream before you issue the DELETE.

## Deleting CF structures

The DELETE STRUCTURE specification requests that an entry for a CF structure be deleted from the System Logger policy. Note that the structure will still be defined to the CFRM policy; this request *only* removes it from the System Logger policy.

There is only one parameter necessary on the DELETE STRUCTURE request:

### NAME

Specifies the name of the CF structure you are deleting from the System Logger policy.

Example 2-9 shows a sample DELETE STRUCTURE request.

*Example 2-9 Sample CF structure delete*

---

```
DATA TYPE(LOGR) REPORT(YES)
DELETE STRUCTURE NAME(SYSTEM_OPERLOG)
```

---

You cannot delete a CF structure from the System Logger policy if there are *any* log stream definitions still referring to it.

## 2.6 Offload processing

As applications write data to a log stream, the interim storage defined for log data begins to fill, eventually reaching or exceeding its high threshold. You specify (in the System Logger policy) high and low thresholds for each log stream to define when System Logger is to begin and end the process of moving, or offloading, log data to offload data sets. It is important for you to understand the way System Logger offloading works so that:

- ▶ You can set appropriate high and low thresholds to control offloading.
- ▶ You understand when, why, and how often offloading will occur.
- ▶ You understand when log data is physically deleted from the log stream.

- ▶ You can plan CF structure, offload data set, and staging data set sizes to control how often offloading occurs.

Offload processing works differently for CF and DASD-only log streams:

- ▶ For a *CF-Structure based log stream*, System Logger offloads data from the CF to offload data sets. Offloading is initiated when *either* of the following occurs:
  - The number of elements in use for the log stream reaches the high threshold.
  - The staging data set for the log stream reaches the high threshold.
  - The last user disconnects from the log stream.

For CF-Structure based log streams, space in interim storage is freed up for reuse incrementally while the offload runs. That is, you do not have to wait until the offload completes before the freed-up space can be used for new IXGWRITE requests.

- ▶ For a DASD-only log stream, System Logger offloads data from local storage buffers to offload data sets. Offloading is initiated when the staging data set reaches the high threshold. Therefore, for a DASD-only log stream, offload processing moves log data from local storage buffers to offload data sets, but the offloading is actually triggered by staging data set usage.

For DASD-only log streams, the space occupied by the log data being offloaded is not freed up until the offload has completed.

There are a number of situations that cause the offload process to be invoked. The situations and the actions System Logger takes are listed in Table 2-5.

Table 2-5 When offload process is invoked

Situation	DASD-only log stream	CF-Structure log stream
Number of elements used OR number of CIs in staging data set used reaches HIGHOFFLOAD	1) Delete <i>all</i> log blocks that have been logically deleted by IXGDELET 2) If LOWOFFLOAD has not been reached, move oldest log blocks to offload data set until LOWOFFLOAD is reached	1) Delete <i>all eligible</i> log blocks that have been logically deleted by IXGDELET 2) If LOWOFFLOAD has not been reached, move oldest log blocks to offload data set until LOWOFFLOAD is reached
Number of entries used in CF structure reaches 90%	Not applicable	1) Delete <i>all</i> log blocks that have been logically deleted by IXGDELET in <i>all</i> log streams in this structure 2) For <i>every</i> log stream in the structure, if LOWOFFLOAD has not been reached, move oldest log blocks to offload data set until LOWOFFLOAD is reached
Structure full	Not applicable	1) Delete <i>all</i> log blocks that have been logically deleted by IXGDELET in <i>all</i> log streams in this structure 2) If LOWOFFLOAD has not been reached, move oldest log blocks to offload data set until LOWOFFLOAD is reached

Situation	DASD-only log stream	CF-Structure log stream
Last connector from a system to a log stream disconnects	All log blocks moved to offload data sets	All log blocks for that log stream moved to offload data sets, up to the newest log block written by the system that just disconnected
Structure rebuild completes and recovery is required	Not applicable	All log blocks for all log streams in this structure moved to offload data sets
Recovery restart	First connection	First connection

For either type of log stream, when offload processing is initiated because the high offload threshold is reached, System Logger begins the process as follows:

1. Delete any eligible log blocks in interim storage that have been marked logically deleted (using IXGDELETE). For a CF-Structure log stream, free the associated interim storage in blocks as the offload proceeds.
2. If the low threshold has not been reached, calculate how much data must be offloaded to reach the low offload threshold.
3. Allocate the offload data set (if it isn't already allocated).
4. Offload the data to the offload data sets, moving the oldest log blocks first. Note that all valid log blocks are moved until the threshold is reached, regardless of which system created them. So, if the offload was initiated because one system disconnected from the log stream, *all* log blocks, from *all* systems, will be moved to the offload data set. Once again, for a CF-Structure log stream, interim storage is freed up as the offload proceeds.
5. For DASD-only log streams, free all interim storage associated with the offloaded log blocks.

Figure 2-11 shows an overview of a two-way sysplex where applications are using CF-Structure based log streams. After some time, the log data is offloaded from the CF structure to offload data sets.

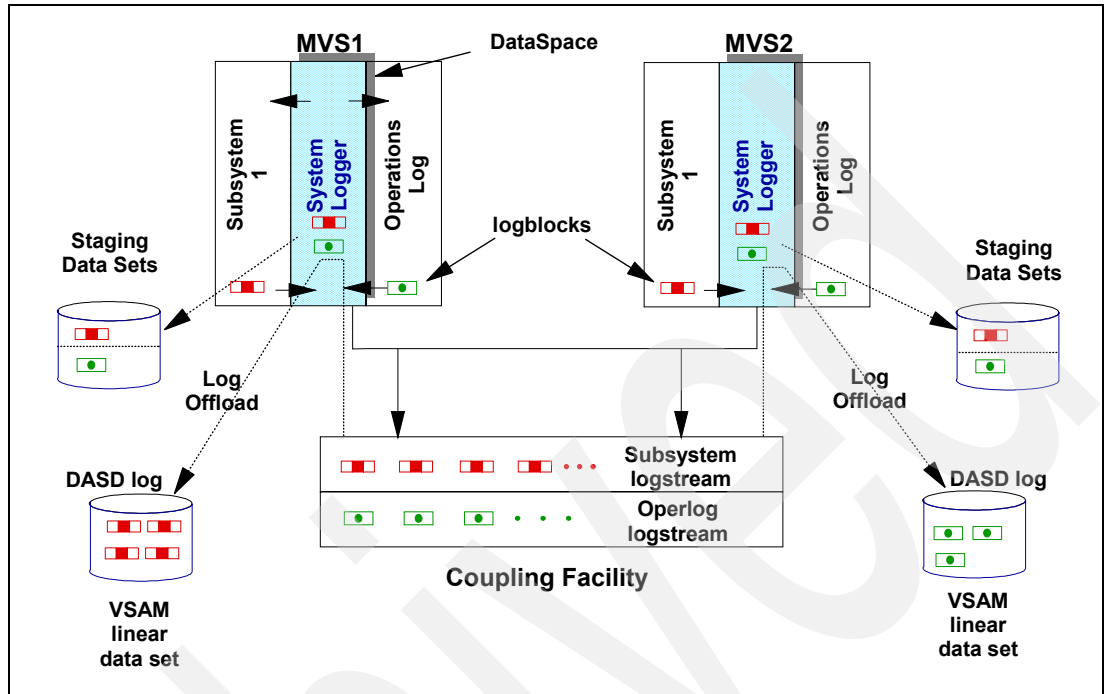


Figure 2-11 Offload overview (example is a CF-Structure based log stream)

Figure 2-12 is a graphical example of the CF structure state before and after the offload, assuming that the high offload threshold is set to 80% and the low offload threshold is set to 20%. Note that new writes can arrive during the offload process, so it is likely that when offload is complete, the storage percentage used will be greater than the defined LOWOFFLOAD value.

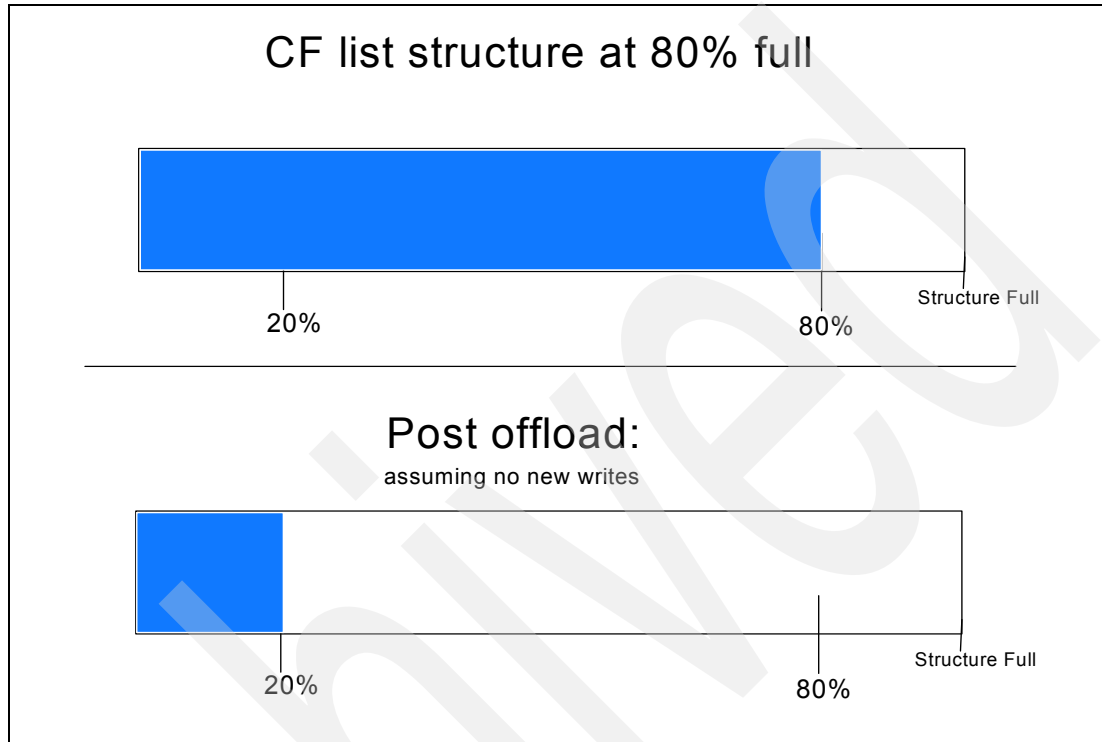


Figure 2-12 Offload from HIGHOFFLOAD to LOWOFFLOAD

As we discussed in previous chapters, the HIGHOFFLOAD and LOWOFFLOAD values allow you to control the offload frequency and the amount of data offloaded. For your System Logger application log streams, you should check the related application chapter in this book for suggested values. However, in general for funnel-type log streams, the default values of 80% HIGHOFFLOAD and 0% LOWOFFLOAD are a good setting. They have the advantage of moving larger chunks of log data to DASD at one time, reducing the number of times that offload processing is invoked for the log stream. Active-type log streams on the other hand might find a HIGHOFFLOAD threshold of 80% and a higher low offload threshold (60%, for example) more desirable as more data will be kept in the CF structure or in local buffers so that it is quickly accessible and you don't waste resources moving data that is about to be deleted to DASD.

You can use SMF reports to tune your offload threshold settings; for more information about this, see 8.6.3, "SMF Type 88 records and IXGRPT1 program" on page 291. In addition, APAR OW13572 added the ability to create a Component Trace record every time an offload process is started—this information can help determine the frequency of offloads for each log stream.



**Important:** We recommend that you do not define your HIGHOFFLOAD value to greater than the default of 80%. If you define a higher HIGHOFFLOAD value, you will be more vulnerable to filling your CF structure or staging data set space for the log stream during sudden bursts of system activity that push the log data above the high threshold. Once CF structure or staging data set space for a log stream is 100% filled, System Logger rejects all write requests for that log stream until the log data can be offloaded to offload data sets.

## 2.6.1 CF-Structure size and offloading

For a CF-Structure based log stream, you can influence the frequency of offloads and the processing overhead offloading entails by appropriate sizing of your CF structures. The larger the CF structure, the less often offload processing will occur, and the larger the amount of data that will be offloaded. This minimizes offloading overhead and keeps more log data available in the CF structure for high performance reads of log data.

For example, if you have a log stream with a high volume of data being written to it, and you have allocated a small CF structure for this log stream, the structure will fill quickly and hit its high threshold, initiating frequent offloading of log data to DASD. This increases system usage, but does not interfere with processing of write requests to the log stream, because write requests can be processed at the same time as offload processing.

However, the CF structure for a log stream should not be larger than the application really requires and should reflect the needs of the rest of the installation for CF space. See “Setting up for and defining CF structures” on page 24 for information about optimal sizing of CF structures for response, throughput, and availability.

If the CF space allocated for a log stream reaches 100% utilization, all write requests against that log stream are rejected until offloading can complete.

## 2.6.2 Staging data set size and offloading

*For CF-Structure based log streams:* For a CF-Structure based log stream, the high offload threshold defined by an installation for the structure space allocated to a log stream applies also to the staging data sets for that log stream. For example, if you define a high threshold of 80% for a log stream, System Logger will begin offload processing when either the CF structure allocated for a log stream *or* a staging data set for the log stream reaches or exceeds 80% usage.

This means that when a staging data set reaches the high threshold, System Logger immediately offloads data from the CF structure to offload data sets, even if the structure usage for the log stream is below the high threshold.

Therefore, if your staging data sets are small in comparison to the CF structure size for a log stream, the staging data sets will keep filling up and System Logger will offload CF log data to DASD frequently. This means that your installation experiences frequent offloading overhead that could affect performance. It also means that you have allocated space in the CF that you will never be able to use (because the data keeps being offloaded before the structure gets anywhere near full).

If your staging data sets are too small, you also run the risk of them filling up completely. If this occurs, System Logger immediately begins offloading the CF log data to offload data sets to harden it. System Logger applications will be unable to log data until System Logger can free up staging data set space.

In general, the sizing guideline for staging data sets is to make them large enough to hold all the log data in the CF structure, allowing for the fact that if there are  $n$  log streams in the structure, each log stream will only be allocated  $1/n$  of the space in the structure. See 8.2.2, “Sizing interim storage” on page 275 for complete information about sizing staging data sets.

*For DASD-only log streams:* For a DASD-only log stream, offloading of log data to offload data sets is always triggered by staging data set usage. System Logger offloads data from local buffers to offload data sets when it hits the high threshold defined for the staging data sets. For example, if you define a high threshold of 80% for a DASD-only log stream, System Logger will begin offload processing when the staging data set space defined for a log stream reaches or exceeds 80% usage.

This means that if you size your staging data sets too small, System Logger will offload log data from local buffers to offload data sets frequently, incurring additional overhead. You might also run the risk of filling the staging data sets up completely. If this occurs, System Logger will immediately begin offloading log data to offload data sets.

**Attention:** APAR OW51854 added the ability for System Logger to monitor offloads and provide a mechanism to interrupt hung offloads. This function is explained further in 7.2, “Offload monitoring” on page 261.

Additional consideration on the offload process can also be found in “The entry-to-element ratio” on page 26.

## 2.7 Deleting log data

One of the most important aspects of running System Logger is understanding how and when log data is deleted. Since log data and offload data set deletion is tied to offload processing, make sure you have first read 2.6, “Offload processing” on page 58.

### 2.7.1 When log data is marked for deletion

Log data is marked for deletion (but not yet physically deleted) based on the use of the IXGDELETE service. This information is then merged with the values used for RETPD and AUTODELETE (see 2.5, “Log streams” on page 22 for parameter usage) to determine when it will be physically deleted:

- ▶ Using RETPD(0) and AUTODELETE(NO): In this case, the only one that can delete data from the log stream is the connector; that is, System Logger will never delete the data itself. Applications should mark log blocks for deletion using the IXGDELETE service as soon as they are no longer needed.
- ▶ Using RETPD(>0) and AUTODELETE(NO): In this case, data must be marked for deletion using the IXGDELETE service *and* the retention period *must* have expired before log data will be deleted. For example, if you specify RETPD(7) in the System Logger policy for a log stream, log blocks will not be deleted for *at least* seven days, even if an IXGDELETE is issued immediately after the log block is written to the log stream. Equally, if seven days pass and an IXGDELETE still has not been issued, the data will not be deleted, and will remain in the log stream until an IXGDELETE is eventually issued.

Note that for data that has been offloaded, System Logger processes the retention period on an offload data set basis. Only after the retention period for *all* the data in the offload data set has expired, is the data set eligible for deletion. Until that time, the data is still accessible, even though its retention period might have expired.

- ▶ Using RETPD(>0) and AUTODELETE(YES): AUTODELETE(YES) means that log data can be physically deleted *either* when the data is marked for deletion by IXGDELET, or when the retention period specified for the log stream expires.

Use care when specifying AUTODELETE(YES). Automatic deletion is designed to speed physical deletion of log data, which can mean deletion of data that an application still needs. Generally speaking, AUTODELETE(YES) should never be used with an application that issues IXGDELETs to delete log records when it no longer requires them.

Note that the default retention period is 0, so the log blocks for log streams specifying AUTODELETE(YES) and RETPD(0), or not specifying any RETPD, will be offloaded to offload data sets and then be eligible for deletion *immediately*.

Some example retention period and autodelete policies can be found in Example 2-10.

**Note:** This retention period is different from a data set retention period on a JCL DD statement. A System Logger retention period applies to the age of the log data, not the data set.

#### Example 2-10 AUTODELETE and RETPD sample policies

##### Example 1. Delete Data After No More Than 3 Days, No Matter What:

For audit log streams, such as the OPERLOG log stream, the installation must often manage how much data is kept in the log stream. Using automatic deletion and a retention period, you can manage the amount of time log data resides in the log stream, cutting down also on the storage consumed by log data in this log stream. For example, let's say that log AUDIT1 requires that data be kept for no more than 3 days and then physically deleted. For AUDIT1, you would specify RETPD(3) and AUTODELETE(YES) to make sure that data is kept for no more than 3 days.

##### Example 2. Keep Data for At Least 30 Days, Even If It Was Marked for Deletion:

Installations may require that their transaction manager log streams retain data for a certain length of time to meet audit requirements. This data must be kept for the retention period, even when it has been marked for deletion via the IXGDELET service. Using automatic deletion and a retention period, you can archive the data for the required length of time without having to move the data out of the log stream. For example, log TRANS1 needs to have data in the log stream for at least 30 days, even if it has been marked for deletion. You can specify RETPD(30) AUTODELETE(NO) to keep the data for the required amount of time. Even if data is marked for deletion via IXGDELET within the 30 day retention period, it is kept in the log stream and can be accessed by a System Logger application using the IXGBRWSE service with VIEW=ALL.

##### Example 3. Keep Data in the Log Stream Until Deletion:

Some log streams may not need to keep log data around after it has been marked for deletion via IXGDELET and do not require that data be retained for a particular length of time. For this kind of a log stream, the default settings of RETPD(0) and AUTODELETE(NO) should be used. System Logger will consider data eligible for physical deletion any time after it has been marked for deletion via IXGDELET.

## 2.7.2 When is my log data or offload data set physically deleted?

When log data is marked for deletion by IXGDELET, System Logger does not physically delete the log data or offload data set until an offload process is initiated (usually as a result of the high threshold being reached). This means that there will often be a delay before eligible offload data sets are physically deleted, since you have to wait for the next offload to start. If offloads are relatively infrequent, there may be a considerable delay before offload data sets that are eligible for deletion are actually deleted.

The point at which offload data sets are physically deleted depends on whether you have specified a RETPD >0 for this log stream. If the log stream is defined with RETPD=0, expired offload data sets<sup>2</sup> will be deleted the next time an offload process is initiated, even if no data is actually moved to an offload data set.

However, if the log stream is defined with RETPD>0, expired offload data sets will not be deleted until the next time System Logger allocates a new offload data set. If the offload data sets are so large that it takes many offloads before the data set is filled (at which point a new offload data set will be allocated), you could potentially have expired data sets hanging around a long time waiting to be physically deleted. And as long as the data sets have not been deleted, they will be holding entries in the directory extent.

Allocation errors can also delay offload data set deletion. If System Logger cannot delete an offload data set because it is currently allocated, it will attempt to delete the offload data set on a subsequent offload when delete processing is again performed. Successive allocation errors may cause a data set to be “orphaned”; an offload data set that System Logger attempted to delete but was unable to. When an offload data set is orphaned, System Logger issues message IXG252I. System Logger frees up the offload data set's entry in the directory extent, meaning that System Logger no longer has any knowledge of that data set—such data sets must be deleted manually. To get a list of orphaned data sets, run an IXCMIAPU report. Because System Logger has removed the entries for the data sets from its inventory, the report job scans the catalog for data sets with names that match the offload data set for that log stream—any such data sets that are found are reported as orphaned data sets. For more information, see “Orphaned Data sets” on page 257.

An example allocation problem where System Logger would be unable to delete the offload data set is demonstrated by Figure 2-13 on page 67. Assume system A initially allocated and offloaded to the first offload data set (LSN.A0000001). System Logger on system A keeps that data set allocated until it has to offload for that particular log stream again (at which point it would allocate the current offload data set). Note that the reason for System Logger requiring SHAREOPTIONS (3 3) is because it uses a SHR ENQ on the offload data sets: so, just because one System Logger has the offload data set allocated does not mean that another System Logger can't write to that data set.

In the meantime, system B maybe have caused several data set switches to occur by offloading so that the current offload data set is now LSN.A0000004. Suppose now that all of the log blocks in data set LSN.A0000001 have been marked for deletion, and system B begins to offload. System Logger will attempt to delete LSN.A0000001, but will fail because system A still has it allocated. System Logger will still go forward and delete other unallocated data sets (LSN.A0000002, and so on) as they become eligible for deletion during offloads.

**Important:** Except for cleaning up orphaned offload data sets, you should *never* delete offload data sets manually. When System Logger is finished with a data set, it will delete it. Deleting an offload data set by any other means could result in lost data. It will also cause problems for System Logger because it will not be able to perform any clean-up and the deleted data sets will still count toward the data set directory limit.

<sup>2</sup> An *expired* offload data set is one in which all the log blocks have reached their RETPD, *and* either an IXGDELET has been issued *or* AUTODELETE(YES) was specified for the log stream.

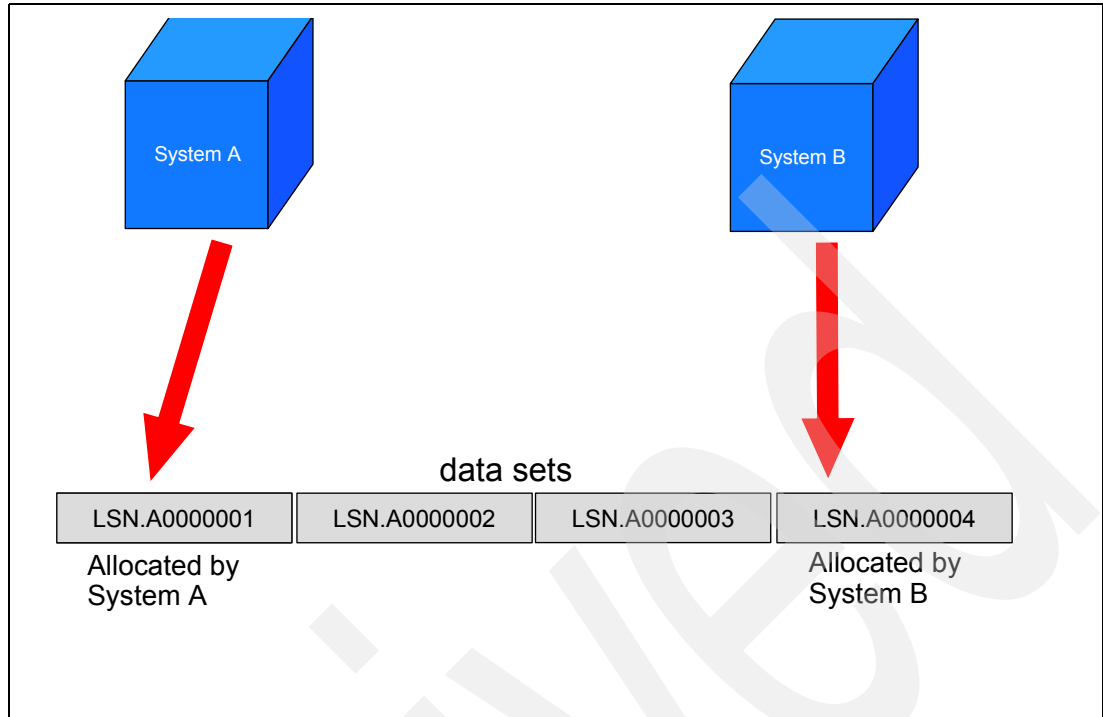


Figure 2-13 Offload data set LSN.A0000001 cannot be deleted

## 2.8 System Logger recovery

In this chapter we discuss how System Logger handles errors or CF structure state changes, how log streams and applications connected to them might be affected, and what actions (if any) should be taken to resolve the issue. As part of this discussion, you should have a basic understanding of how System Logger works to provide failure independence, as it allows for greater recoverability of data and how System Logger processing attempts to recover both log stream types. We'll close out this chapter by discussing how System Logger handles some other common errors.

### 2.8.1 Failure independence

System Logger provides the ability to protect your log data from loss due to the failure of a system component; that is, System Logger tries to provide an environment by duplexing log data such that no single points of failure exist for that log stream. A single point of failure is an environment where one failure can result in the simultaneous loss of both copies of the log data currently resident in interim storage. For example, if the CF resides in the same CPC as the System Logger that is connected to it, the CF structure for a log stream and the local buffer copy of the data would both be lost if that CPC were to fail.

When your configuration contains a single point of failure, you can safeguard your data by telling System Logger to duplex log data to staging data sets (System Logger will do this depending on the environment the system is in and the log stream definition; see 2.5, "Log streams" on page 22 for more information).

In Figure 2-14 on page 68, a log stream has been defined with STG\_DUPLEX=YES and DUPLEXMODE=COND. There are two systems connected to the log stream. System Sys 1, resides in the same CPC as the CF containing the log stream, so the data that Sys 1 writes to

the log stream is duplexed to a staging data set because there is a single point of failure between the system and the CF. For example, a machine error might require a reset of CPC1, which would result in the loss of both the CF and Sys 1 with its local buffer copy of the CF log data if it was not duplexed to staging data sets. On the other hand, there is no single point of failure between Sys 2 (which resides in a different CPC) and the CF. In this case, System Logger on Sys 2 will keep the duplex copy of the log data in its local buffers. If the CF fails, the data is still available in the local buffers, and if the CPC or Sys 2 fails, the data is still available in the CF. So, you can see that the single definition results in two instances of System Logger behaving differently because of their different environments.

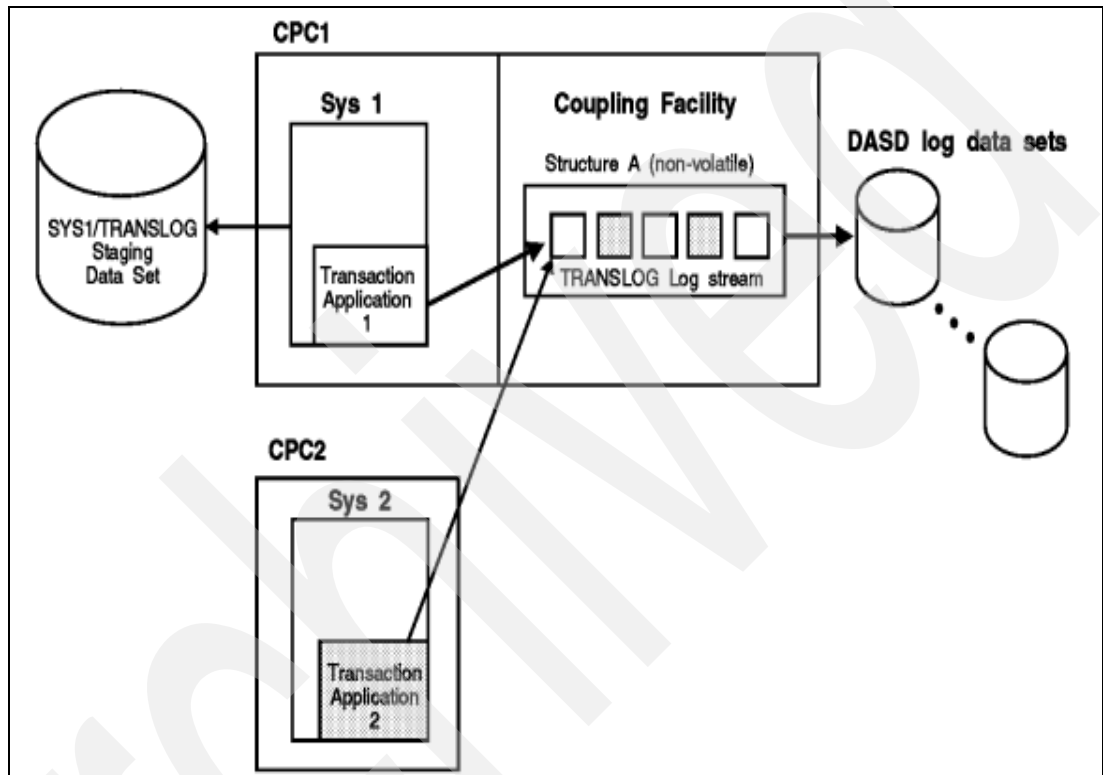


Figure 2-14 A single point of failure exists between Sys1 and the CF

A single point of failure can also exist for a CF structure using System-Managed Duplexing where one failure can result in the simultaneous loss of both structure instances; this would be if the two CFs are in the same CPC, or if both CFs are volatile. The two structure instances in this environment are failure-dependent, meaning a single failure will cause all the structure data to be lost, even though it is in two copies of the structure. Since a single point of failure can cause the loss of all the log data in this case, System Logger will continue to safeguard the data by using one of its own duplexing methods.

System Logger evaluates a connection for a single point of failure based on the location of the CF and its status of volatile or non-volatile. The CF can reside in one of the following configurations:

- ▶ The CF executes in an LPAR in the same CPC as a system that is connected to a log stream resident in the CF.
- ▶ The CF is in a stand-alone CPC and therefore does not share the CPC with a system with a connection to the log stream.

System Logger uses the following rules for determining whether a log stream connection contains a single point of failure:

- ▶ When a CF containing a LOGR structure shares a CPC with a system that is connected to that structure, each active connection between the system and the CF contains a single point of failure, regardless of the volatility state of the CF. The connections have a single point of failure because if the CPC should fail, CF resident log data will be lost when both the system, with local buffers, and the CF fail with the CPC.
- ▶ When a CF (or the composite view of a duplex-mode CF structure) is separate from the system connected to the associated log stream, the volatility status of the CF determines whether the connection is vulnerable to a single point of failure:
  - If the CF (or the composite view of a structure using System-Managed Duplexing) is non-volatile, the connection is failure independent.
  - If the CF (or the composite view of a structure using System-Managed Duplexing) is volatile, the connection is vulnerable to a single point of failure. A power failure could affect both the systems, with its local copy of log data, and the CF.

A connection is failure independent when it is from a system on a separate CPC from the non-volatile CF to which it is connected.

If a connection contains a single point of failure and STG\_DUPLEX(YES) DUPLEXMODE(COND) has been specified in the log stream definition, System Logger will duplex CF log data to staging data sets for that connection. In the event of a CF or CPC failure, System Logger can retrieve lost CF log data from the staging data set. See 2.5.1, “CF-Structure based log streams” on page 23 and “Duplexing log data for CF-Structure based log streams” on page 42 for information about using the STG\_DUPLEX and DUPLEXMODE keywords.

**Note:** If you specify STG\_DUPLEX(NO), make sure your application is not vulnerable to data loss that might occur if your environment contains a single point of failure.

## 2.8.2 Failure recovery

Now that you have an understanding of how System Logger works to provide a failure independent environment, we can move on to the question “What happens if System Logger, the system or the sysplex fails?” Recovery is different for the two types of log stream (CF-Structure based and DASD-only) and there are a few new operations that we will discuss as well, such as structure rebuild. We’ll start at the highest level of System Logger recovery (system/sysplex level recovery) and work down to the details of CF-Structure based log stream recovery; we’ll then discuss DASD-only log stream recovery as a separate topic.

### System level recovery

When System Logger fails, it will attempt to restart itself, providing you have installed the PTF for APAR OW53349. If it is unable to successfully restart itself after a few attempts, or you have issued the FORCE IXGLOGR,ARM command to terminate the System Logger address space, restarting will require use of the S IXGLOGRS command (for more information about these commands, see Chapter 7, “Logger operations” on page 253.

During startup, System Logger runs through a series of operations for all CF-Structure based log streams to attempt to recover and cleanup any failed connections; and to ensure all data is valid. DASD-only log stream recovery is not handled at this time so we’ll discuss it as a separate topic.

As part of this process, System Logger will issue connect requests for each CF-Structure based log stream that the system was connected to at the time it failed (that is, for each CF-Structure based log stream that there now exists a failed-persistent connection to). For each of these log streams, System Logger will, if necessary, clean up each failed persistent connection to it (regardless of the system it originated on); System Logger will then attempt the same operation for *each log stream connected to the same CF structure* (that is, perform peer recovery, which is covered in the next topic). Note that recovery will attempt to offload any log data in interim storage to offload data sets and then clean up the failed connection. If recovery fails for some reason, the failed persistent connection will be kept to ensure data is not lost.

Let us look at Figure 2-15 for an example of a recovery operation. Assume both system A and B failed, and system A is IPLed first. Since system A was connected to log stream A when the system failed, recovery would be attempted for log stream A. As part of this recovery operation, System Logger would cleanup the connections from system A *and* system B because both systems will have failed-persistent connections to the log stream. System Logger would then try to recover any failed connectors for all the other log streams residing in structure A; here, log stream B only has one connection, from system B. However, since log stream B is in the same structure as log stream A (for which System Logger on system A initiated recovery) the failed persistent connection from system B to log stream B would be recovered as well. As a result of these recovery operations (assuming they are successful) there would be no connections to structure A, and all log data in log streams A and B has been offloaded to offload data sets.

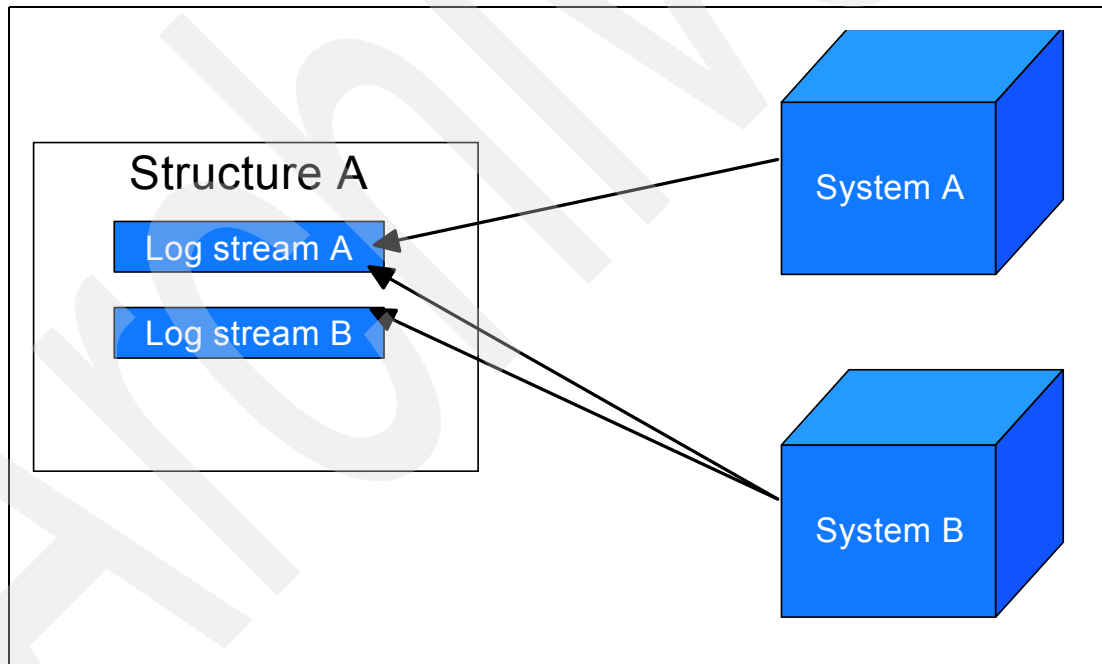


Figure 2-15 Structure A has peer connectors

**Peer and same system log stream recovery**

Going back to Figure 2-15, let us now assume only system A failed. Since system B is connected to the same CF structure as system A, system B will be informed that System A has failed and will attempt to recover and cleanup the connection from system A to log stream A; this is an example of peer recovery. Peer recovery does not take place for DASD-only log streams.



In order for peer recovery to take place, the CF structure must be connected to by more than one system, and at least one system must still be available after the failure. Note that when a system containing a System Logger application fails, another System Logger will try to safeguard the CF log data for the failed system, either by offloading it to offload data sets, or by ensuring that the log data is secure in a persistent duplex-mode structure. This recovery processing is done by a peer connector, which is another system in the sysplex with a connection to the same CF structure as the failing system. In general, when you set up your logging configuration, you should try to make sure that each log stream structure is accessed by more than one system.

If there is no peer connection available to perform recovery for a failed system, recovery is delayed until either the failing system re-IPLs or another system connects to a log stream in the same CF structure to which the failing system was connected. In Figure 2-16, for example, peer recovery can be performed on structure B. However, recovery for structure A log data is delayed until either Sys 1 re-IPLs, or another system connects to a log stream in that structure.

Every time a new first connection to a log stream structure is established from a particular system, System Logger examines the log stream to determine if recovery is needed. If this is also the first connection to this structure, System Logger will check if recovery is needed for any other log streams in the structure.

**Note:** To ensure peer recovery, a peer connector is only needed to the same *structure* from another system in the sysplex; it does not have to be to the same *log stream*.

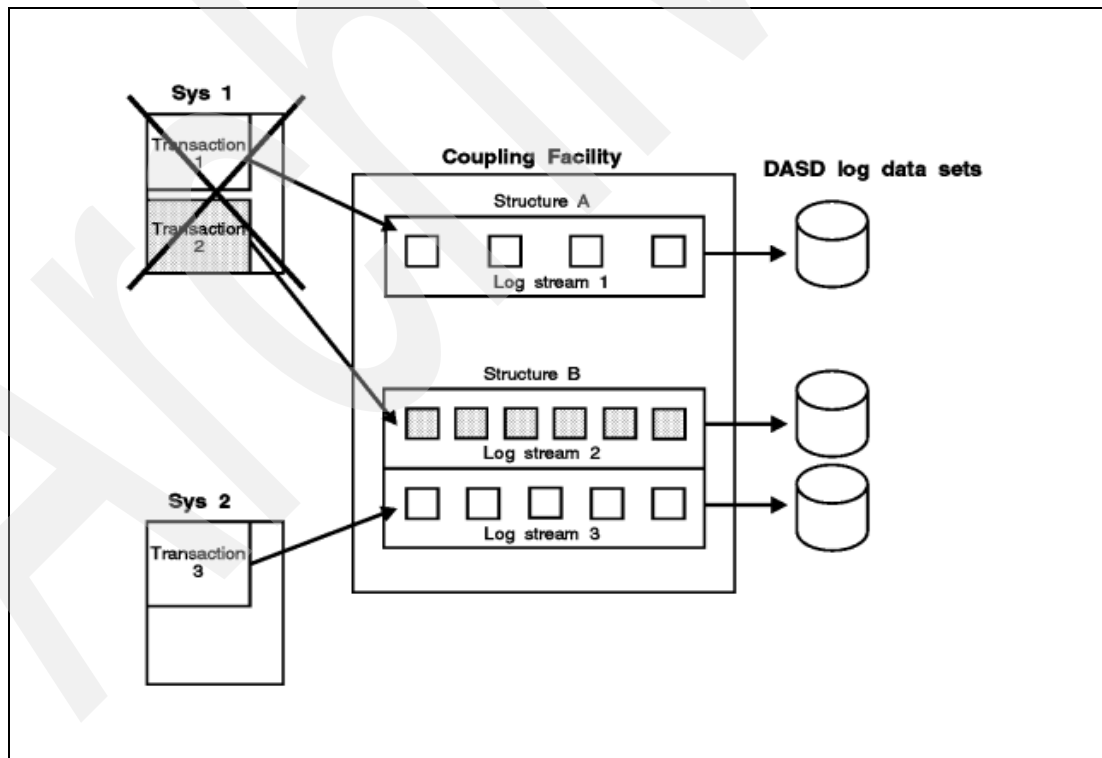


Figure 2-16 Recovery delayed for some data

To avoid having a structure with no peer connectors, we recommend one of the following:

- ▶ Make sure that another log stream with connectors from other systems maps to the same CF structure.
- ▶ Write a dummy application to run on another system that will connect to the same log stream or CF structure to which the failed system was connected.

### ***What if both the system and the CF fail?***

If the CF containing the log stream structure and the CPC containing System Logger both fail (because of a power failure, for example), all log data for that log stream that was still in interim storage will be lost if staging data sets were not being used for that log stream. If staging data sets exist, System Logger uses them to recover the log data. If no staging data sets were in use when both the system and the CF fails, any log streams affected are marked as damaged. This is an example of a situation where choosing not to use staging data sets results in System Logger not being able to prevent a single point of failure from existing.

### **DASD-only log stream recovery**

Like a CF-Structure based log stream, a DASD-only log stream can be affected by a system or System Logger failure. However, because DASD-only log streams are always duplexed to DASD, it is not necessary to go through the same recovery process that is required for CF-Structure based log streams, where System Logger attempts to move the log data to a non-volatile medium as quickly as possible. Also, because DASD-only log streams can only be accessed by one system at a time, the concept of peer recovery does not apply.

Recovery for a DASD-only log stream only takes place when an application reconnects to the log stream. As part of connect processing, System Logger reads log data from the staging data set (associated with the last connection to the log stream) into the local buffers of the current connecting system. This allows the application to control recovery, by selecting which system they wish to have reconnect to the log stream and when. Note that for another system to connect to the log stream and perform recovery, the staging data sets must reside on devices accessible by both systems.

## **2.8.3 Other recovery processes and errors**

As we are discussing System Logger recovery, there are a few other internal recovery processes that you should be aware of. In this section, we discuss how System Logger reacts to CF structure state changes and common errors.

### **Structure state changes**

The following CF problems can occur, resulting in rebuild processing for the structure:

- Damage to or failure of the CF structure.
- Loss of connectivity to a CF.
- A CF becomes volatile.

In this topic, we'll discuss each of these situations individually, as well as giving some insight into System Logger processing during a structure rebuild.

Note that this section only applies to CF-Structure based log streams and *not* DASD-only log streams.

### ***Structure rebuild***

Structure rebuild from a System Logger perspective is the process by which a new instance of a CF structure is created and repopulated after System Logger has been notified by XES that a rebuild has been initiated (either by an operator command, structure failure, or state

change). Because System Logger provides support for user-managed rebuilds, system-managed rebuild is not used to rebuild the structures unless there are no active connections to the structure.

You can read more about structure rebuild processing from an overall system perspective in Chapter 5.5 of *z/OS Sysplex Services Guide*, SA22-7617.

CF structure rebuilds are intended for planned reconfiguration and recovery scenarios. The next few sections describe the most common events that trigger a structure rebuild.

As we mentioned, System Logger responds to rebuild events communicated to it by XES. System Logger has its own internal exits to notify connectors of the current state of the process. While the rebuild is in progress, System Logger rejects any service requests against the log stream—it is up to the connector to understand these reason codes and retry their requests. The new structure instance is then repopulated and the connectors that were actively connected to the original structure instance are connected to the new instance. Log data that was resident in interim storage will be copied from the duplex copy (that is, the local buffers or staging data sets, never from the old structure) to the new structure instance. Once the structure has been rebuilt (rebuilding is complete), the system deallocates the original structure and connectors can use the new structure. In some cases, System Logger will initiate a directed offload to move all log data in the structure at this point.

Note that while the structure is duplexed using System-Managed Duplexing, rebuild is not possible. If the structure needs to be rebuilt (for example, to clear a POLICY CHANGE PENDING condition), the structure must be reverted to simplex mode first.

**Note:** If the new structure instance cannot be successfully repopulated for any reason, and System Logger does not have connectivity to the original structure instance, staging data sets will be created to recover the log data to, *regardless* of the log stream's STG\_DUPLEX parameter.

### ***Damage to or failure of the CF structure***

If a structure was being duplexed using System-Managed Duplexing and a CF failure or structure damage occurs to only one instance of the structure, XES will automatically revert the structure back to simplex mode. System Logger will only be notified of the mode switch change and will not be aware of any CF failure or damage to the structure. See “When the CF structure duplex/simplex mode changes” on page 76 for information about how System Logger handles mode switches for the structure.

If the structure was in simplex mode and the failure or damage occurs to the only instance of the CF structure, all systems connected to the CF structure detect the failure. The first system whose System Logger component detects the failure initiates the structure rebuild process. The structure rebuild process results in the recovery of one or more of the affected CF structure's log streams. All the systems in the sysplex that are connected to the CF structure participate in the process of rebuilding the log streams in a new CF structure. We discussed CF structure rebuilds in “Structure rebuild” on page 72.

### ***Loss of connectivity to the CF structure***

If a CF structure was in duplex mode and a loss of connectivity occurs to only one instance of the CF structure (due to a hardware link failure), XES will automatically switch the structure from duplex mode to simplex mode. System Logger will only be notified of the mode switch change and will not be aware of any loss of connectivity to the structure. See “When the CF structure duplex/simplex mode changes” on page 76 for how System Logger handles mode switches for the CF structure.

For a simplex mode structure, System Logger detects the loss of connectivity to the single instance of the structure. Then, based on the rebuild threshold specified, if any, in the structure definition in the CFRM policy, the system that lost connectivity may initiate a rebuild for the structure.

If XES cannot allocate a new structure instance in a CF that can be connected to from all the systems using that structure, System Logger does one of the following, depending on whether the system or systems that cannot connect to the new CF structure were using staging data sets:

- ▶ If the system was using staging data sets, the rebuild process continues and the CF log data for the system is recovered from the staging data sets.
- ▶ If the system was not using staging data sets, the rebuild process is stopped. The systems go back to using the source structure. The log stream will be available on the systems that have connectivity to the source CF structure; it will be unavailable on those that do not have connectivity.

### ***When the CF structure volatility state changes***

The following CF volatility state changes can occur, which may result in different reactions by System Logger:

- ▶ A CF becomes volatile, or if the structure is duplexed using System-Managed Duplexing, both CFs could become volatile.
- ▶ A CF becomes non-volatile, or if the structure is duplexed using System-Managed Duplexing, both CFs could become non-volatile.

System Logger's behavior to the state change depends on:

- ▶ The type of state change.
- ▶ The current duplexing environment for the log streams in the affected structures.
- ▶ The log stream and CF structure duplexing specifications from the LOGR CDS and the CFRM policy.

Some of the environments may result in rebuild processing for the structure. For more information about rebuild processing see "Structure rebuild" on page 72.

### **When a CF becomes volatile**

If a CF changes to the volatile state, the System Logger on each system using the CF structure is notified.

For structures that are in simplex mode, a rebuild of the structure is initiated so that the log data can be moved to a non-volatile CF. During rebuild processing, System Logger rejects any service requests (connect, and so on).

If there is not a CF structure available in a non-volatile CF, System Logger will still rebuild the structure to a new volatile CF. System Logger may then change the way it duplexes CF data because the volatile CF constitutes a single point of failure:

- ▶ For log streams defined with STG\_DUPLEX=YES, System Logger will begin duplexing data to staging data sets, if it was not already doing so.
- ▶ For log streams defined with STG\_DUPLEX=NO, System Logger will keep on duplexing data to local buffers on each system.

For structures that are duplexed using System-Managed Duplexing, structure rebuild is not possible because there are already two copies of the structure. However, System Logger may

change whether it will duplex the log data and the way it duplexes the data since the new composite CF volatility state constitutes a single point of failure.

- ▶ If the initial environment had a non-volatile composite structure CF state, but there was a failure-dependent relationship between the connecting system and the composite structure view, the log stream was considered to be in a single point of failure environment and still is after the state change, so System Logger makes no changes to the duplexed method.
- ▶ If the initial environment had a non-volatile composite structure CF state and there was a failure-independent connection between the connecting system and the composite structure view, but there was a failure-dependent relationship between the two CF structure instances, the log stream was not considered to be in a single point of failure environment, but would be after the CF state changed to volatile. System Logger would continue to duplex the log data, but might change the duplexing method.

For log streams defined with STG\_DUPLEX=YES, System Logger will begin duplexing data to staging data sets, if they were not already in use.

- ▶ If the initial environment had a non-volatile composite structure CF state, there was a failure-independent connection between the connecting system and the composite structure view, and there was a failure-independent relationship between the two CF structure instances, the log stream was not considered to be in a single point of failure environment, but would be after the CF state changed to volatile:
  - For log streams defined with LOGGERDUPLEX(UNCOND), System Logger will continue to duplex the log data. However, for log streams defined with STG\_DUPLEX=YES, System Logger will begin duplexing data to staging data sets, if they were not already in use.
  - For log streams defined with LOGGERDUPLEX(COND), System Logger will first offload the log data from the CF structure, then begin duplexing any new log data written to the log stream.
  - For log streams defined with STG\_DUPLEX=NO, System Logger will begin duplexing data to local buffers.
  - For log streams defined with STG\_DUPLEX=YES, System Logger will begin duplexing data to staging data sets.

### **A CF Becomes Non-Volatile**

If a CF changes to a non-volatile state, the System Logger on each system using the CF structure is notified.

For simplex mode structures, System Logger may change the way it duplexes CF data because the change to a non-volatile CF may have removed the single point of failure condition.

- ▶ If the initial environment had a volatile structure CF state and there was a failure-*dependent* relationship between the connecting system and the CF structure, the log stream was considered to be in a single point of failure environment and continues to be so after the state change, so System Logger makes no changes to the duplexing method.
- ▶ If the initial environment had a volatile structure CF state and there was a failure-*independent* relationship between the connecting system and the CF structure, System Logger may then change the way it duplexes CF data because the non-volatile CF no longer constitutes a single point of failure.

System Logger will duplex the log data using local buffers, unless the log stream definition specified STG\_DUPLEX(YES) DUPLEXMODE(UNCOND).

For duplex-mode structures, System Logger may also change the way it duplexes CF data because the change to a non-volatile CF may have removed the single point of failure condition.

- ▶ If the initial environment had a volatile composite structure CF state but there was a failure-dependent relationship between the connecting system and the composite structure view, the log stream was considered to be in a single point of failure environment and continues to be so after the state change (because of the failure-dependent connection), so System Logger makes no changes to the duplexing method.
- ▶ If the initial environment had a volatile composite structure CF state and there was a failure-independent connection between the connecting system and the composite structure view, but there was a failure-dependent relationship between the two structure instances, the log stream was considered to be in a single point of failure environment, but would not be after the CF state changes to non-volatile. System Logger would continue to duplex the log data, but might change the duplexing method.
  - The data will be duplexed by System Logger using local buffers, unless staging data sets were specifically requested by having STG\_DUPLEX(YES) DUPLEXMODE(UNCOND) on the log stream definition.
- ▶ If the initial environment had a volatile composite structure CF state, there was a failure-independent connection between the connecting system and the composite structure view, and there was a failure-independent relationship between the two structure instances, the log stream was considered to be in a single point of failure environment, but would not be after the CF state changed to non-volatile.
  - For log streams defined with LOGGERDUPLEX(UNCOND), System Logger will continue to duplex the log data.
  - The log data will be duplexed by System Logger using local buffers, unless staging data sets were specifically requested by having STG\_DUPLEX(YES) DUPLEXMODE(UNCOND) on the log stream definition.
  - For log streams defined with LOGGERDUPLEX(COND), System Logger will stop providing its own duplexing because the system-managed duplexed structure provides sufficient back-up capability for the log data.

### ***When the CF structure duplex/simplex mode changes***

This section applies to CF log streams only. If a CF structure changes modes, the System Logger on each system using the CF structure is notified.

System Logger's behavior after a structure switches from simplex-mode to duplex-mode or vice versa depends on whether a single point of failure environment exists after the mode switch and the duplexing specifications for the structure and the log streams within the structure.

It is possible for System Logger to change from using local buffers to staging data sets or to change from using staging data sets to local buffers. It is also possible for System Logger to start providing its own duplexing of the log data or stop duplexing the log data after the mode switch.

“Duplexing log data for CF-Structure based log streams” on page 42 shows how System Logger decides what storage medium to duplex log data to based on the environment System Logger is operating in and the log stream definition.

## 2.9 An introduction to SMF88 records

SMF Type 88 records contain information about usage of a log stream data for a system in a sysplex. Using the data from the Type 88 records can help you tune and perform capacity planning on your log streams.

Type 88 records summarize all a log stream's activity on a single system, as long as at least one address space is connected to the log stream on that system. If no System Logger write activity is performed against the log stream during a particular SMF interval, a record is produced showing zero for the various System Logger activity total fields.

The Type 88 record is produced for all log streams connected at the expiration of the SMF global recording interval. The creation of a Type 88 record is also triggered by the disconnection of the last connector to the log stream on that system.

There are some things you must remember about the Type 88 records however:

- ▶ The Type 88 records only show activity from one system. If a log stream is used by more than one system, you must merge the information from all systems to get a comprehensive view of the use of that log stream
- ▶ There are a number of subtype records produced by System Logger, but only one of them, the one dealing at the log stream level, has an IBM-provided formatter. Therefore, to determine what is happening at the structure level for structures containing more than one log stream, you need to merge the information from all log streams in that structure, from all systems in the sysplex.

In 7.9.2, “SMF88” on page 270 you can find a discussion on how to set up for collecting Type 88 records. After the Type 88 records are collected, they can be sorted and analyzed using a tool; IBM ships one of these tools as a Samplib member called IXGRPT1. We'll discuss using IXGRPT1 in 7.9.3, “IXGRPT1” on page 271.

IXGRPT1 provides one of the few ways to effectively tune your log streams for better performance. However, interpreting the report that is produced by the IXGRPT1 program is not a trivial task. In 8.6.3, “SMF Type 88 records and IXGRPT1 program” on page 291 you can find a discussion on using the Type 88 data to tune System Logger performance as well as highlighting particular fields of interest in some of the application chapters.

## 2.10 Recommended service

In this section we discuss maintaining the System Logger component and what documentation should be collected when a problem occurs to speed up problem diagnosis.

### 2.10.1 Collecting documentation for problem diagnosis

Depending on the problem, the following seven methods may be used to collect the documentation needed to diagnose a System Logger problem. This documentation should be collected before contacting the IBM support center; for more information, see the System Logger chapter in *z/OS MVS Diagnosis: Reference, GA22-7588*: amongst other things, this chapter tells you how to set up a component trace for System Logger and how to interpret the output from an IXCMIAPI report.

1. Obtain a dump of the System Logger address space and any address spaces connected to the log stream you are experiencing a problem with. Refer to “DUMP command parmlib member” on page 270 for sample on the dump keywords.

2. Use the D LOGGER command to display the following information:
  - IXGLOGR address space status
    - **D LOGGER,ST**
  - Log stream, CF structure, and connection information.
    - **D LOGGER,L**
    - **D LOGGER,STR**
    - **D LOGGER,C**
  - Sysplex status for log streams.
    - **D LOGGER,C,SYSPLEX**
  - Specifics for DASD-only log streams.
    - **D LOGGER,C,DASDONLY**
    - **D LOGGER,L,DASDONLY**
3. Use IDCAMS to print the offload data sets for a log stream using the JCL shown in Example 2-11.

*Example 2-11 IDCAMS job to print offload data sets*

---

```
//IDCAM1 JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*-----*/
/* PRINT contents of System Logger offload data set */
/*-----*/
/*
//PRINTIT EXEC PGM=IDCAM5
//SYSPRINT DD SYSOUT=*
//OFFLOAD DD DSN=hlq.xxx.A00000yyy.DATA,DUSP=SHR
//SYSIN DD *
PRINT INFILE(OFFLOAD)
/*
```

**Notes:**

- hlq is IXGLOGR by default, unless HLQ(hlq) is specified when the log stream is defined
  - xxxx is the defined log stream name
  - A0000yyy is the generation number LLQ created by System Logger
-



4. Obtain offload data set characteristics. You can use the sample JCL in Example 2-12 to display the characteristics of the data set you are dumping.

*Example 2-12 IDCAMS sample JCL to show DS characteristics*

---

```
//IDCAM2 JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//PRINTIT EXEC PGM=IDCAM2
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  LISTCAT ALIAS ALL
  LISTCAT ALIAS ALL CAT('SROCAT.CATALOG')
  LISTCAT ENT('USER.CATALOG.NAME') ALL CAT('USER.CATALOG.NAME')
  LISTCAT LVL('HLQ_NAME') ALL
/*
```

This job will:

- Display all alias names specified in the master catalog, along with the associated user catalog for each alias.
  - Display all alias names defined in a specified catalog.
  - Display the contents of a user catalog and the volume on which it resides.
  - Display all information related to data sets with a particular high level qualifier.
- 

5. Obtain a System Logger inventory detailed listing by running an IXCMIAPU LIST report as shown in Example 2-13.

*Example 2-13 Sample IXCMIAPU report job*

---

```
//LOGRRPT JOB (999,POK),'LOGR POLICY',CLASS=A,REGION=4M,
//          MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  LIST LOGSTREAM NAME(logstreamname) DETAIL(YES)
/*
```

---

6. If you suspect that the LOGR CDS is corrupted, print its contents using the job shown in Example 2-14.

*Example 2-14 Sample job to dump the LOGR CDS*

---

```
//DUMPCDS JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=SYSUID
//*****/
//* RUN ADDRDSU to dump off the LOGR Couple Data set */
//*****/
//STEP1 EXEC PGM=ADDRDSU,REGION=4M
//SYSPRINT DD SYSOUT=*
//DD1 DD DISP=SHR,VOL=SER=xxxxxx,UNIT=SYSDA
//SYSIN DD *
  PRINT DATASET(logr.couple.dataset) INDDNAME(DD1) TOL(ENQF)
/*
```

---

**Note:** As we discussed in 2.5, “Log streams” on page 22, specifying DIAG(YES) on the log stream definition will provide additional documentation in certain error cases that could not be otherwise collected.

Set a SLIP trap. Your IBM support representative will provide you with the information you need to set the appropriate SLIP for the problem you are experiencing. See *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589 for more information about this.

### **2.10.2 Recommended maintenance policy**

To avoid rediscovering problems that already have a fix available, it is important that you keep your System Logger component up to date. Therefore, we recommend you apply all PTFs for:

- ▶ HIPER APARs
- ▶ APARs with the keyword LOGRSERVICE in the APAR description
- ▶ APARs with the keyword CFDUPLEX in the APAR description

## DFSMStvs and Logger

This chapter discusses how DFSMStvs (also known as Transactional VSAM) uses System Logger.

In this chapter we provide:

- ▶ A basic description on how Transactional VSAM works
- ▶ How many log streams are required and where and how to define them
- ▶ Coupling facility configuration versus DASD-only description
- ▶ Operational and performance information
- ▶ Recovery information

### 3.1 Function description

Before VSAM Record Level sharing, recoverable files could only be opened by a single CICS region, and all requests for file updates had to be routed to that lone region.

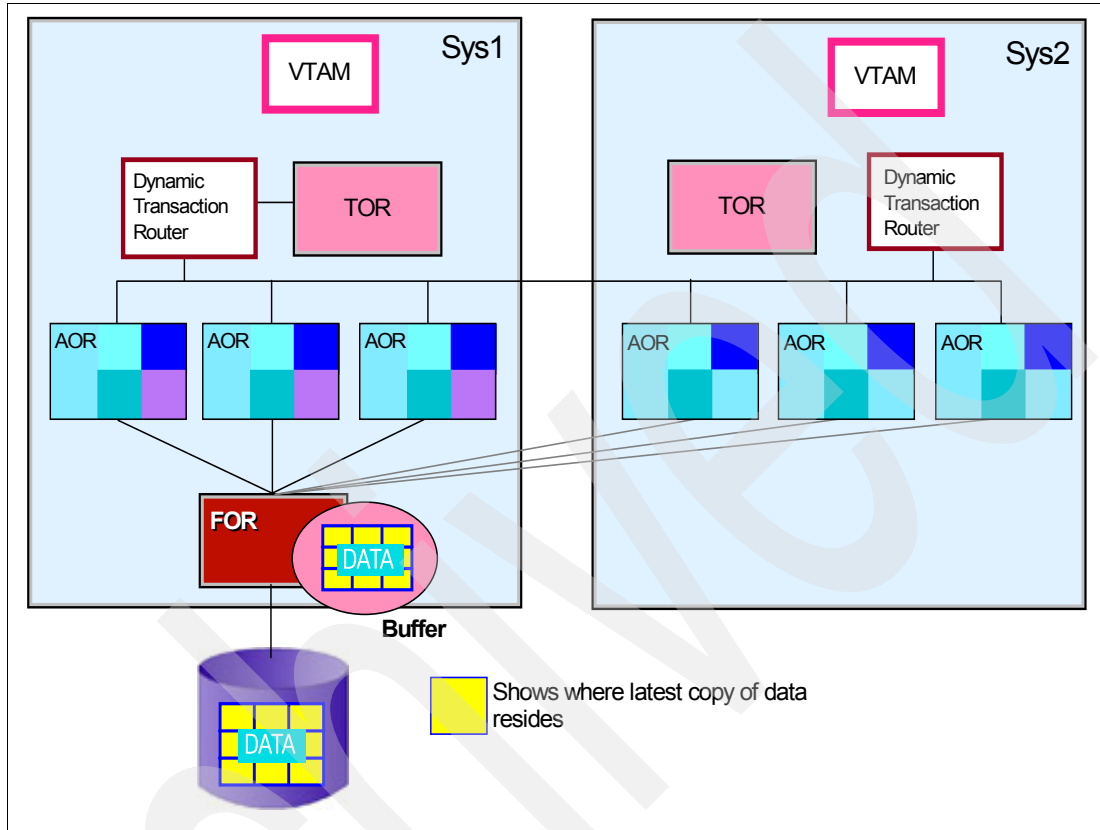


Figure 3-1 VSAM data sharing before Record Level Sharing (RLS)

With the advent of VSAM Record Level Sharing, multiple CICS regions could access recoverable VSAM data sets simultaneously; in this configuration, the SMSVSAM address space provided the locking functions to ensure proper serialization while the CICS regions performed the logging. Since the logging was encapsulated within CICS, batch jobs could not update recoverable VSAM data sets while the data sets were open for update in the CICS regions.

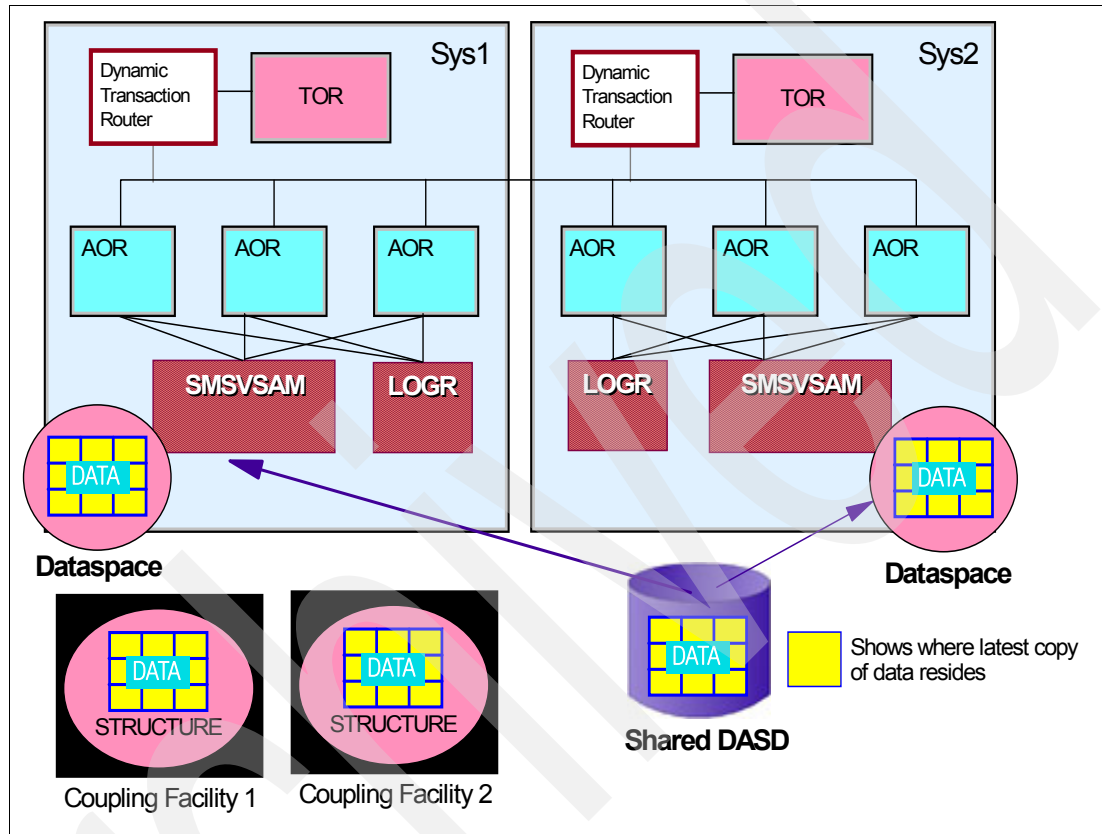


Figure 3-2 CICS/VSAM Record Level Sharing

Transactional VSAM extends the data sharing capability of VSAM RLS to batch. Multiple batch jobs can now update recoverable VSAM data sets while the data sets are open for update to the CICS regions. SMSVSAM still provides the locking, and the CICS regions still provide the logging for the updates made to the data sets by CICS transactions, and now DFSMStvs provides the logging for updates made by batch jobs.

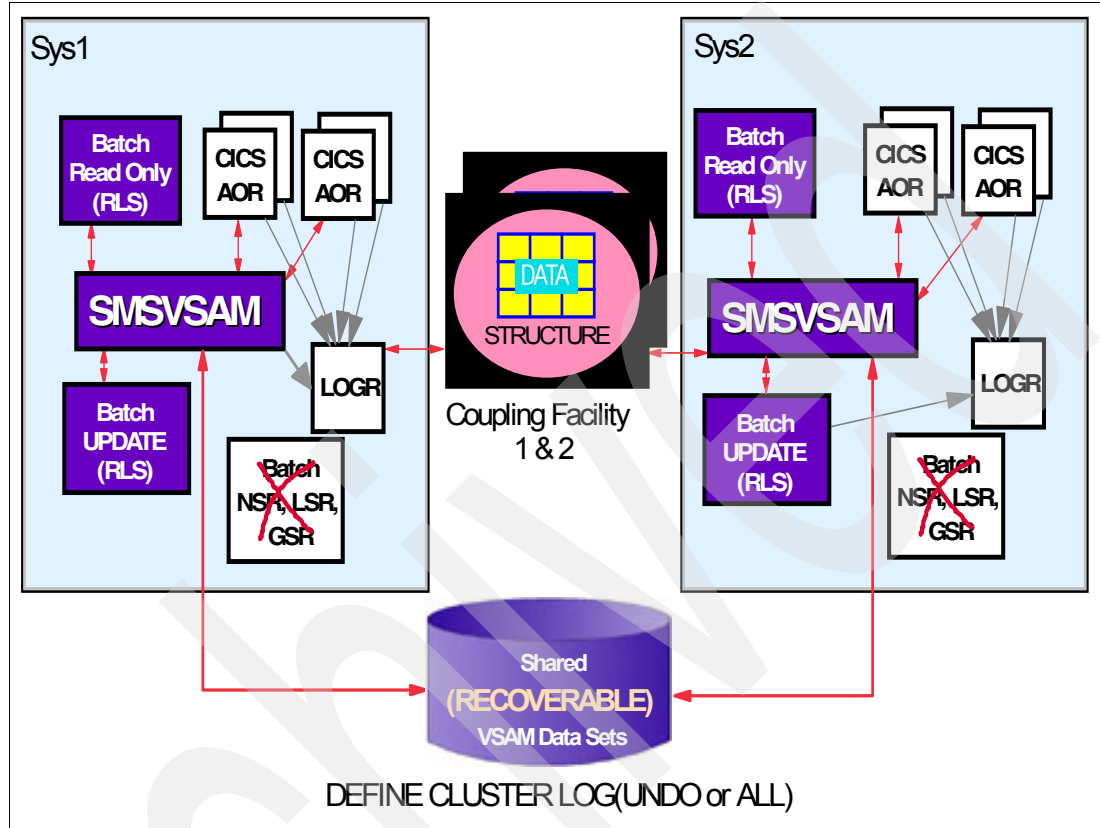


Figure 3-3 Transactional VSAM and CICS

Transactional VSAM provides two-phase commit for updates to VSAM files by registering with Resource Recovery Services (RRS). Since Transactional VSAM uses RRS, all the documentation about this function tends to use the RRS terminology for recovery services. For that reason, what is normally referred to as a Unit of Work (UOW) in CICS manuals is called a Unit of Recovery (UR) in DFSMStvs documentation. A UR is the set of changes made by a particular task until the task invokes either the RRS Commit or RRS Backout, at which time all the changes are either committed to the VSAM data sets or backed out. The commit can either be explicit or implicit (at the end of the step). If the batch job does not issue any explicit commits, the entire step is considered a single UR. Any committed changes are not backed out if there is a subsequent Backout call; only those changes made since the previous commit are backed out.

To keep track of all the before-copy images that might have to be restored to the VSAM data sets (which is what happens when a UR indicates that it wants to “backout” the changes since the last commit point), Transactional VSAM needs to record the before-copy images.

Under normal circumstances, there is only one copy of Transactional VSAM on each image within the sysplex (for exceptions to the practice, refer to *DFSMStvs Administration Guide*, GC26-7483), and each instance of Transactional VSAM records the before-copy images for URs executing on its system into a unique undo log. For performance reasons, the one logical undo log consists of two physical log streams, referred to as IGWLOG.SYSLOG and

IGWSHUNT.SHUNTLOG. As the names imply, IGWLOG.SYSLOG is the primary repository for before-copy images while IGWSHUNT.SHUNTLOG is the repository for information about shunted URs.

### 3.1.1 Log stream usage

Transactional VSAM directly uses three types of log streams: the undo log, forward-recovery and the log-of-logs.

#### The Undo log

Each instance of TVS (and normally there is only one instance on each image within the sysplex on which you want to access recoverable VSAM data sets from batch) requires its own unique undo log. The undo log consists of two physical log streams, referred to as IGWLOG.SYSLOG and IGWSHUNT.SHUNTLOG. Each instance of Transactional VSAM has a unique name (for more information about how to define this name, see 3.2.1, “Subsystem definitions” on page 90), and this name is incorporated into the log stream names to ensure that each instance of Transactional VSAM gets a unique set of Undo log stream. The complete names are:

IGW&tvname.IGWLOG.SYSLOG and IGW&tvname.IGWSHUNT.SHUNTLOG, where &tvname is the unique name of the instance of Transactional VSAM.

DFSMStvs writes the ‘before’ image of a record in a recoverable VSAM data set to the IGWLOG.SYSLOG log stream. If an in-flight UR issues a backout or the UR abends, DFSMStvs uses the data in the Undo log stream to restore the original contents of the record to the VSAM data set.

DFSMStvs moves data from IGWLOG.SYSLOG to IGWSHUNT.SHUNTLOG in two circumstances:

- ▶ If a UR updates a record in a recoverable VSAM data set, but then fails to make an update to another record in recoverable VSAM data set within one activity keypoint, DFSMStvs moves the chain of log records for that UR since the last sync point (or the start of the UR) from IGWLOG.SYSLOG to IGWSHUNT.SHUNTLOG.
- ▶ If a UR goes into INDOUBT status, when a resource manager fails between the two phases of the two-phase commit.

These two log streams are active log streams. This means that DFSMStvs manages the content of the log streams and deletes data from them as it is no longer required, and this process is referred to as log tail management. For more information about the difference between active and funnel-type log streams, refer to 1.1.1, “How System Logger is used” on page 3.

If you run Transactional VSAM on three systems within your sysplex, you need to define at least six log streams: two for each of the three instances with a total of three IGWLOG.SYSLOG log streams and three IGWSHUNT.SHUNTLOG log streams.

The undo log streams are required. Transactional VSAM attempts to connect to the undo log streams when it initializes (either during the IPL of the system or in response to a V SMS,TRANVSAM(tvname),ENABLE command). If the connection to either one of the log streams is not successful, Transactional VSAM does not initialize.

## The forward recovery log streams

When you use IDCAMS to define a data set, the LOG keyword determines the “recoverability” of the data set. There are three possible values for LOG:

▶ LOG(NONE)

The data set is non-recoverable. Uncommitted changes cannot be backed out, and the data set cannot be reconstructed in case of a failure.

▶ LOG(UNDO)

The data set is recoverable. Uncommitted changes may be backed out (as either a result of an explicit RRS Backout request or the step ending abnormally); however, the data set cannot be reconstructed in case of a failure.

▶ LOG(ALL)

The data set is recoverable. Uncommitted changes may be backed out. In addition, Transactional VSAM records the after-images (the updated version of the record or an indication that a record was deleted) in the forward recover log stream. If the data set fails, a separate product like CICS/VR can rebuild the data set by recalling the last back-up version of the data set and reapplying all the changes recorded in the forward recovery log streams.

If you code LOG(ALL), you must also specify the name of the forward recovery log stream using the LOGSTREAMID keyword.

If you define a data set with a forward recovery log stream, both CICS TS and Transactional VSAM write to the same log stream. Since both parties contribute to the same log stream, all the updates to the data set are used to rebuild a failed data set.

The forward recovery log streams are funnel-type log streams. This means that Transactional VSAM and CICS TS simply add data to the log streams; they never delete any data from these log streams.

The forward recovery log streams are required for data sets defined with LOG(ALL). If Transactional VSAM cannot connect to the log stream associated with a given data set, the OPEN for the data set fails. Each instance of Transactional VSAM in the sysplex connects to the forward recovery log stream associated with a data set as part of the OPEN processing. Similarly, when the last data set associated with a given log stream is closed, Transactional VSAM disconnects from the forward recovery log stream.

CICS TS connects to and disconnects from the forward recovery log streams in a similar manner (refer to 5.1.1, “Log stream usage” on page 148), however, there is one difference. The recoverable data sets tend to remain open during the life of the CICS region. With Transactional VSAM, the connections to the forward recovery log streams might be made and broken more frequently depending on the data set pattern of reference and the number of batch jobs that run concurrently. If in your installation, the data sets are only used by batch jobs (and are not opened to CICS regions) and the forward recovery log streams are defined to use staging data sets, you might see message IGW8381, indicating a temporary delay while the staging data set is formatted, frequently as the connections come and go every time the data set opened and closed by the batch jobs.

## The log-of-logs

The log-of-logs log stream is optional; you do not need to define or use a log-of-logs log stream. The log-of-logs log stream houses all the tie-up records, which indicate when data sets are opened and closed. The forward recovery product (like CICS/VR) can use this information to reconstruct a damaged data set more quickly.



If you choose to use a log-of-logs log stream, you should define the same log stream to both Transactional VSAM and CICS TS. For more information about defining the log-of-logs log stream to Transactional VSAM, refer to 3.2.1, “Subsystem definitions” on page 90. For details on how to define the log-of-logs log stream to CICS TS, see 5.2.2, “System Logger definitions” on page 154.

The log-of-logs log stream is a funnel-type log stream. This means that Transactional VSAM and CICS TS simply add data to the log stream; they never delete any data from this log stream.

The log-of-log log stream is optional; Transactional VSAM connects to the log-of-logs when it initializes (either during the IPL or in response to the `V SMS,TRANVSAM(tvsnam),ENABLE` command), and if the connection to the log stream is unsuccessful, Transactional VSAM continues to initialize without a log-of-logs.

### **Indirect use of log streams**

Transactional VSAM registers with RRS and uses RRS services. Since RRS maintains its own log streams, Transactional VSAM indirectly contributes to the RRS-related log streams. For more information about RRS, see 6.4, “Resource Recovery Services” on page 228.

## **3.1.2 Criticality/persistence of data**

The three types of log streams that Transactional VSAM uses each has its own level of criticality, ranging from low through medium to high. At the low end is the log-of-logs log stream. Since this log stream only contains data that makes the recovery of a failed data set quicker, this log stream (and the data within it) can be lost without adversely effecting overall system performance or functionality.

The next level of criticality is for the forward recovery log streams. If a forward recoverable data set should fail, the data in the forward recovery log stream is vital to the restoration of that data set. Without all the forward recovery images, CICS/VR or similar product cannot rebuild the data set from the last back up copy of the failed data set. For this reason, if a forward recovery log stream fails (or is inaccessible or simply does not exist), Transactional VSAM prohibits updates against the data set and CLOSEs the data set. The installation then must take a fresh back-up and rebuild the log stream.

Even though the data in the undo log stream is only maintained while it might be needed to perform a UR backout, the two log streams that comprise the undo log (IGWLOG.SYSLOG and IGWSHUNT.SHUNTLOG) are absolutely critical to the operation of Transactional VSAM. Without those two log streams, Transactional VSAM cannot function and terminates.

## **3.1.3 Log streams sizing**

Log stream data exists in temporary or interim storage, and in permanent or hardened storage. This section discusses sizing of the interim storage in order to achieve the best log stream and overall system performance.

There are many variables which must be taken into account when sizing a log stream, making it difficult to get it perfect the first time around. Once an initial size is determined, the log stream performance must be monitored using the SMF Type 88 records produced by the System Logger, and adjustments made, as required.

Understanding the volume and rate data is written is critical when sizing the log stream interim storage.

## Interim storage and duplexing

Interim storage can be viewed as the location where the log stream data can be accessed quickly - without I/O to long term DASD (offload data sets). For a CF-Structure log stream, interim storage is usually a CF structure. For a DASD-only log stream, interim storage is contained in local storage buffers. Local storage buffers are data spaces associated with the System Logger address space, IXGLOGR.

Each time a log block is written to a log stream, the System Logger automatically creates a duplicate copy of the data. For DASD-only log streams, the data is written to the local buffers (data space) and duplexed to the staging data set. For a CF-Structure log stream, the data will also be duplexed to the local buffers unless a staging data set is being used or if CF-structure duplexing is being used. A staging data set is when the CF is volatile or is in a failure dependent configuration and STG\_DUPLEX(YES) DUPLEXMODE(COND) has been specified in the log stream definitions, or if STG\_DUPLEX(YES) DUPLEXMODE(UNCOND) was specified.

A CF is considered volatile when it no longer has a battery backup. A CF is considered to be in a failure-dependent environment when the CF LPAR and the z/OS LPAR are in the same physical box; that is, there is a single point of failure. The concept of failure independence as it relates to System Logger is discussed in more detail in 2.8.1, "Failure independence" on page 67.

The staging data set used to duplex a CF-Structure log stream must be large enough to hold the largest amount of data which can be contained in the CF log stream interim storage (that is, the part of the log stream that is currently in the structure in the CF).

With the exception of CF-Structure log streams duplexed to either staging data sets or CF-managed duplexed structures, the data space will contain the duplex copy of all data resident in interim storage for all active log streams in the z/OS image.

To optimize use of the data space, and reduce the potential impact on storage use, residency time of the data should be kept to minimum. For the TVS Undo log (IGWLOG and IGWSHUNT), the data should be retained in interim storage until the associated UOW commits or backs out. For forward-recovery log streams, all data is offloaded to the offload data sets so the interim storage (CF structure or staging data set) should be sized to provide minimal residency time.

Offload data sets contain data which has been offloaded from interim storage. Data is offloaded from interim storage as part of the offload process, usually initiated when the HIGHOFFLOAD threshold is reached. Offload data sets may be subsequently archived using a product like DFSMSHsm. For a complete description of offload processing, refer to 2.6, "Offload processing" on page 58.

## Sizing interim storage

**Note:** Remember, sizing of a log stream is an iterative process. Unfortunately, it is difficult to get it perfect the first time.

Once a starting size has been identified, the SMF Type 88 records should be used to tune the log stream based on the activity during the key processing intervals.

When calculating the size of interim storage for a log stream, the following log stream characteristics are important:

The number of IXGWRITE requests issued during a specified time interval.

- ▶ TVS requests the System Logger to record information on a log stream using an IXGWRITE call. The number of IXGWRITE requests is found in the SMF Type 88 records for the log stream (SMF88LWB or IXGWRITES).
- ▶ Rate of IXGWRITE calls - LGSWRITES/interval in seconds.
- ▶ Number of bytes written in the interval. The number of bytes written is found in the SMF88 (SMF88SWB) records written for the log stream.
- ▶ Number of bytes written per IXGWRITE or average buffer size. Bytes written divided by the number of write requests (LGSBYTES/LGSWRITES). Reported as the average buffer size in the SMF 88 reports produced with IXGRPT1.
- ▶ HIGHOFFLOAD percentage. This determines at what point the offload starts and also determines how much space is left between when the offload starts and when the log stream fills.
- ▶ The number of log stream deletes for IGWLOG and IGWSHUNT found in the SMF88 (SMF88SII and SNF88SAI). This is a reflection of the number of times DFSMStvs performed log tail management. Log tail management is the process where DFSMSTVS issues an IXGDELETE call to indicate records which are no longer required for recovery and may be deleted. Log tail management reduces the size required for interim storage without having to offload the data.
- ▶ Number of offloads during interval. Provided in the SMF Type 88 records (SMF88EO).

Each TVS Undo log stream requires enough storage to hold the *data written in an Activity KeyPoint (AKP) interval + the duration of the longest UOW + a buffer of at least 25%*. Structures should be large enough to hold the sum of the data written for all connected log streams in the interval.

### Undo log streams

The methodology used to perform an initial sizing for the Undo log streams would follow the process listed below. Using this method may result in oversizing the log stream which should then be reduced based on the activity reported in the SMF Type 88 records. It is also difficult to size the Undo log stream initially since the all the sizing considerations are usually done against the current batch window and later on batch might spread over a different schedule because of the TVS enhancement changing completely the planned requirement. For this reason, it is even more important, after an initial consideration, to evaluate the configuration through the SMF88 records.

As of now, there are no tools to help size the Undo log stream. A suggested approach is to use the SMF64 at CLOSE time to obtain the number of writes against the data set. Multiple the number for the average recordsize and divide for the Activity Key Point: this should give you an idea of how much storage you should need in the interim media for this data set. If you sum all the data sets open in a certain interval, this should give you an idea of how much total storage you need for your Undo log stream.

### Forward Recovery log streams

The forward recovery log streams are funnel-type log streams (all data is offloaded). Oversizing forward recovery log streams increases the working set of the System Logger and may result in overall system paging problems. Usually, an installation already has these log stream because of the CICS. You probably need to monitor and re-evaluate the size with SMF88 since the activity might increase against these log stream.

For details, see CICS section, 5.3.5, "Log stream sizing" on page 175.

## 3.2 Definitions

All log streams (and the structures used by CF-Structure log streams) must be defined in the System Logger policy. Some of the log streams are hard-coded in nature and are only defined in the System Logger policy; others log streams can have variable names and are specified in parmlib or in the catalog.

### 3.2.1 Subsystem definitions

Below is information about where you need define the log stream names to your TVS subsystem.

#### The Undo Log

Each instance of Transactional VSAM requires two unique log streams, and the name of the particular instance of Transactional VSAM is incorporated into the log stream names. In parmlib member IGDSMSxx, you associate the name of the Transactional VSAM instance for a given system using two keywords: SYSTEM and TVSNAM. There are three different ways to code these keywords: a unique IGDSMSxx member for each system in the sysplex, a shared IGDSMSxx member for all systems in the sysplex without system symbols, and a shared IGDSMSxx member for all systems in the sysplex using system symbols. For the following examples, assume that we have three systems in the sysplex called #@\$1, #@\$2 and #@\$3 and that we want to run Transactional VSAM instance 001, 002 and 003 on them (Transactional VSAM instance names must be numeric in the range from 001 to 999).

#### **Unique IGDSMSxx member for each system in the sysplex**

With this method, you require three unique IGDSMSxx members, one for each of the three systems. In the member used on system #@\$1, you would code as shown in Example 3-1.

*Example 3-1 Sample IGDSMSxx member for system #@\$1*

---

```
SYSTEM(#@$1)
TVSNAM(001)
```

---

Similarly, for the member used on #@\$2, you would code as shown in Example 3-2.

*Example 3-2 Sample IGDSMSxx member for system #@\$2*

---

```
SYSTEM(#@$2)
TVSNAM(002)
```

---

Finally, for the member used on #@\$3, you would code as shown in Example 3-3.

*Example 3-3 Sample IGDSMSxx member for system #@\$3*

---

```
SYSTEM(#@$3)
TVSNAM(003)
```

---

With this methods, you need to remember to create a new parmlib member every time you add a system in the sysplex.

### **Shared IGDSMSxx member without system symbols**

With this method, you only need one IGDSMSxx, but you have to define all the systems in the sysplex in the member. For this example, you would code as shown in Example 3-4.

*Example 3-4 Sample IGDSMSxx member for all system without using system symbols*

---

```
SYSTEM(@$1, @$2, @$3)
TVSNAME(001,002,003)
```

---

**Note:** The values for these keywords are positional and you must ensure that you specify the values for SYSTEM in the same order as TVSNAME. In addition, if you add (or remove) a system to the sysplex, you have to update this parmlib member accordingly.

### **Shared IGDSMSxx member with system symbols**

If you use system symbols, you can code a single IGDSMSxx member that does not need to be updated whenever you add additional members to the sysplex. First you must define a symbol that gets a unique value for each system in the sysplex. In this example, we have symbol &SYSID1 which gets the value 1 on system @\$1, a 2 on @\$2 and 3 on @\$3. Now in the IGDSMSxx you code as shown in Example 3-5.

*Example 3-5 Sample IGDSMSxx member using system symbols*

---

```
SYSTEM(&SYSNAME.)
TVSNAME(&SYSID1.)
```

---

With this method, when you add a system to the sysplex, you only need to ensure that the variable &SYSID1 (defined in IEASYMxx) gets a unique value.

### **The forward recovery log streams**

When you use IDCAMS to DEFINE or ALTER a data set with LOG(ALL), you must also specify the name of the forward recoverylog stream with the LOGSTREAMNAME keyword. Each data set does not need a unique forward recovery log stream; multiple data sets can share the same forward recovery log stream. In this case, usually they are grouped by application.

### **The log-of-logs**

The log stream used for the log-of-logs is defined in two places. For Transactional VSAM, you define the log-of-logs in the IGDSMSxx parmlib member using the LOG\_OF\_LOGS keywords. Refer to 5.2.1, “Subsystem definitions” on page 151 for details on defining the log-of-logs log stream to CICS TS.

### **Activity keypoint frequency (AKP in IGDSMSxx)**

The activity keypoint frequency value specifies the number of write operations to the undo log stream buffer before an activity keypoint is taken. It does not reflect how much data is held in the buffer prior to it being written to IGWLOG.SYSLOG. A keypoint is a snapshot of in-flight URs in the system at that time. The activity keypoint interval is defined as the amount of time between two sequential activity keypoints.

During the activity keypoint, DFSMStvs initiates the log tail trimming for IGWLOG.SYSLOG and IGWSHUNT.SHUNTLOG. After determining the oldest record that is still needed to perform UR backout (this is called the history point), DFSMStvs issues an IXGDELET request to delete all log blocks older than the history point from the log streams. The data is then physically deleted by System Logger during the next offload performed for each log stream.

Refer to 2.7, “Deleting log data” on page 64 for details on the relationship between the logical deletion and physical deletion during an offload.

If you set AKP too high, DFSMStvs performs log-tail trimming less frequently, which results in more data in the log stream. A larger amount of data in the log stream results in an increase in storage usage in the System Logger address space along with an increase in the restart time should DFSMStvs fail.

If you set AKP too low, DFSMStvs performs log-tail trimming more frequently, which might result in the shunting of URs (moving the data associated with a UR from the IGWLOG.SYSLOG to IGWLOG.SHUNTLOG log streams). If the UR ends a little while after the shunting, the shunting was performed needlessly.

The recommendation is to take the default value for this keyword.

### 3.2.2 Logger definitions

DFSMStvs can use either Coupling Facility log streams or DASD-only log streams; it all depends on your configuration and needs. If you are only running Transactional VSAM on one system, then you can use DASD-only log streams for all the log streams. However, given that all the connections to a DASD-only log stream must come from the same system, if you plan on using Transactional VSAM on two or more systems in the system, then you **MUST** define all the forward recovery log streams and the log-of-logs log streams as Coupling Facility log streams; however, even in a multi-system environment, you can keep the undo log (IGWLOG.SYSLOG and IGWSHUNT.SHUNTLOG) as DASD-only log streams (as each instance of Transactional VSAM connects to just its log streams).

#### **Coupling Facility log streams or DASD-only log streams for Undo logs**

There are items to consider when deciding whether to use Coupling Facility log streams or DASD-only log streams for the IGWLOG.SYSLOG and IGWSHUNT.SHUNT log streams. Positive attributes of a Coupling Facility log stream include better performance (if they are properly tuned). On the other hand, if you are constrained for space in the Coupling Facilities, or your Coupling Facilities are failure dependent with your z/OS images (for a discussion on failure dependent and independent connections, refer to 2.8.1, “Failure independence” on page 67), in which case you should use Staging Data Sets anyway, you should consider using DASD-only log streams.

### 3.2.3 Coupling Facility log streams

You can use Coupling Facility log streams for all the log streams used by Transactional VSAM. Here are a few examples for defining each type of log stream where we highlight the important keywords and their definitions.

**Note:** When defining a Coupling Facility log stream, you need to ensure that the structure for the log stream is defined both in the CFRM Policy and the System Logger Policy. The order does not matter (you can update the CFRM policy and then the System Logger Policy, or the other way around). However, both definitions must be made **BEFORE** you attempt to use the desired log stream.

## Defining Coupling Facility log streams

Example 3-6 contains a sample job to update the CFRM Policy with a structures to be used by System Logger for IGWLOG.SYSLOG and IGWSHUNT.SHUNTLOG log streams.

*Example 3-6 Defining structures in the CFRM policy*

```
//DEFCFRM JOB CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID.
//CFRMPOL EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(CFRM) REPOR(YES)
DEFINE POLICY NAME(CFRMPOL1) REPLACE(YES)
DEFINE STRUCTURE NAME(LOG_IGWLOG_001)
  INITSIZE(5120)
  SIZE(10240)
  PREFLIST(cfname1,cfname2)
  ALLOWAUTOALT(NO)
  DUPLEX(ALLOWED)
  FULLTHRESHOLD(0)
DEFINE STRUCTURE NAME(LOG_IGWSHUNT_001)
  INITSIZE(5120)
  SIZE(10240)
  PREFLIST(cfname1,cfname2)
  ALLOWAUTOALT(NO)
  DUPLEX(ALLOWED)
  FULLTHRESHOLD(0)
STRUCTURE NAME(LOG_CICSVR_001)
  INITSIZE(5120)
  SIZE(10240)
  PREFLIST(cfname1,cfname2)
  ALLOWAUTOALT(NO)
  DUPLEX(ALLOWED)
  FULLTHRESHOLD(0)
STRUCTURE NAME(LOG_LOGOFLOGS)
  INITSIZE(2560)
  SIZE(5120)
  PREFLIST(cfname1,cfname2)
  ALLOWAUTOALT(NO)
  DUPLEX(ALLOWED)
  FULLTHRESHOLD(0)
/*
```

**Note:** After running the job to update the CFRM Policy, you must issue the z/OS command to activate the new policy. In this example that is:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=CFRMPOL1
```

When using Coupling Facility log streams, there are several general recommendations:

### ***INITSIZE and SIZE***

The value coded for SIZE should not be more than twice the value coded for INITSIZE. When the structure is allocated, the Coupling Facility Control Code (CFCC) reserves enough control space to accommodate the largest structure possible. If SIZE is much greater than INITSIZE, the portion of control space, which is not usable, system logger does not leave much usable space for System Logger in the structure.

## **PREFLIST**

Eliminate a single point of failure by always having at least two Coupling Facilities. If you only have one Coupling Facility (or if you have multiple Coupling Facilities by only specify one in the PREFLIST for a given structure), if that Coupling Facility needs to come down (for example, to apply a micro-code update to pick-up a newer version of the CFCC), then the structure cannot be rebuilt to an alternate Coupling Facility. By having a second Coupling Facility and specifying at least two Coupling Facilities in the PREFLIST, you allow the structure to be rebuilt from one Coupling Facility to another (either automatically by the operating system or in response to an operator command).

## **ALLOWAUTOALT(NO)**

Since System Logger deletes data from the structure once the log block has been hardened to disk or deleted, the structures used by System Logger may appear to XES as good candidates for an automatic reduction in size. However, the smaller structure size adverse effects System Logger performance. To prevent XES from automatically altering the size of System Logger structures, code ALLOWAUTOALT(NO)

## **FULLTHRESHOLD(0)**

System Logger manages the space within the structure, and at times goes over the default value of 80% structure full. Instead of using the XES messages to determine the structure use, you should use the SMF 88 records and the output from IXGRPT1 or IXGRPT1J to analyze the log stream performance. Therefore, we recommend coding FULLTHRESHOLD(0) to disable the XES monitoring.

In addition to defining the structures in the CFRM policy, you must also define the structures and the log streams in the System Logger Policy. Example 3-7 provides a sample for updating the System Logger Policy.

### *Example 3-7 Defining Coupling Facility log streams in the logger policy*

---

```
//CFBASED JOB CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID.  
//LOGRPOL EXEC PGM=IXCMIAPU  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
DATA TYPE(LOGR) REPORT(NO)  
DEFINE STRUCTURE NAME(LOG_IGWLOG_001)  
LOGSNUM(10)  
AVGBUFSIZE(4096)  
MAXBUFSIZE(64000)  
DEFINE STRUCTURE NAME(LOG_IGWSHUNT_001)  
LOGSNUM(10)  
AVGBUFSIZE(4096)  
MAXBUFSIZE(64000)  
DEFINE STRUCTURE NAME(LOG_CICSVR_001)  
LOGSNUM(10)  
AVGBUFSIZE(4096)  
MAXBUFSIZE(64000)  
DEFINE STRUCTURE NAME(LOG_LOGOFLOGS)  
LOGSNUM(1)  
AVGBUFSIZE(4096)  
MAXBUFSIZE(64000)  
DEFINE LOGSTREAM NAME(IGWTV001.IGWLOG.SYSLOG)  
AUTODELETE(NO)  
RETPD(0)  
DIAG(YES)  
HIGHOFFLOAD(80)  
LOGGERDUPLEX(COND)  
LOWOFFLOAD(60)
```



```

LS_DATACLAS(LOGR24K)
LS_SIZE(2000)
OFFLOADRECALL(NO)
STG_DUPLEX(YES)
DUPLEXMODE(COND)
STG_DATACLAS(LOGR4K)
STG_SIZE(0)
STRUCTNAME(LOG_IGWLOG_001)
LIST LOGSTREAM NAME(IGWTV001.IGWLOG.SYSLOG) DETAIL(YES)
DEFINE LOGSTREAM NAME(IGWTV001.IGWSHUNT.SHUNTLOG)
  AUTODELETE(NO)
  RETPD(0)
  DIAG(YES)
  HIGHOFFLOAD(80)
  LOGGERDUPLEX(COND)
  LOWOFFLOAD(0)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(2000)
  OFFLOADRECALL(NO)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  STG_DATACLAS(LOGR4K)
  STG_SIZE(0)
  STRUCTNAME(LOG_IGWSHUNT_001)
LIST LOGSTREAM NAME(IGWTV001.IGWSHUNT.SHUNTLOG)
DEFINE LOGSTREAM NAME(CICSVR.LOG001)
  AUTODELETE(NO)
  RETPD(0)
  DIAG(YES)
  HIGHOFFLOAD(80)
  LOGGERDUPLEX(COND)
  LOWOFFLOAD(0)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(2000)
  OFFLOADRECALL(NO)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  STG_DATACLAS(LOGR4K)
  STG_SIZE(0)
  STRUCTNAME(LOG_CICSVR_001)
LIST LOGSTREAM NAME(CICSVR.LOG001) DETAIL(YES)
DEFINE LOGSTREAM NAME(CICSVR.LGOFLOGS)
  AUTODELETE(NO)
  RETPD(0)
  DIAG(YES)
  HIGHOFFLOAD(80)
  LOGGERDUPLEX(COND)
  LOWOFFLOAD(0)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(9000)
  OFFLOADRECALL(NO)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  STG_DATACLAS(LOGR4K)
  STG_SIZE(0)
  STRUCTNAME(LOG_LOGOFLOGS)
LIST LOGSTREAM NAME(CICSVR.LGOFLOGS) DETAIL(YES)

```

/\*

**Note:** Before you can reference a structure in a log stream definition in the System Logger Policy, you MUST first define the structure. If you try to define a Coupling Facility log stream before defining the associated structure, the job fails with:

```
IXG018E STRUCTURE strname DOES NOT EXIST
```

```
IXG002E LOGR POLICY PROCESSING ENDED WITH RETCODE=00000008 RSNCODE=00000827
```

Here are the general recommendations for defining the Coupling Facility structures for the Undo logs in the System Logger Policy.

### **Defining a separate structure for IGWLOG and IGWSHUNT**

When using Coupling Facility log streams, you can either use a dedicated structure for each log stream or you can group several log streams onto one structure. (Refer to “How many CF structures you need” on page 25 for details on how to best group log streams in the same structure). If you do group multiple log streams into one structure, you should only place log streams with similar characteristics (arrival rate of log blocks, average size of log blocks and maximum size of log blocks) on the same structure. Since IGWLOG and IGWSHUNT have such different characteristics (IGWLOG is written to frequently while IGWSHUNT is rarely updated), we recommend that you use separate structures for the IGWLOG and IGWSHUNT log streams.

### **Structure related keywords**

Following is a description of the keywords used to define the structure for the log stream(s) in the LOGR policy:

#### ***LOGSNUM(10)***

You should not place more than 10 log streams in the same structure as a larger number adversely effects the System Logger performance when the structure needs to be rebuilt. (Refer to “How many CF structures you need” on page 25 for additional considerations).

#### ***AVGBUFSIZE(4096)***

Specifies the average size of the buffer DFSMStvs passes to the System Logger when it issues an IXGWRITE. For a CF-based log stream, this value, along with the value specified for MAXBUFSIZE, determine the initial entry-to-element ratio for the structure when it is first allocated. MAXBUFSIZE determines the size of the entries: a value of X or more (I need to verify this value) results in an entry size of 512, while any value less than X results in an element size of 256. System logger then calculates how many elements are needed to accommodate the average buffer size, as specified in AVGBUFSIZE, and uses that ratio to set the initial entry-to-element ratio.

When System Logger first became available, it was vital to code these two values correctly as System Logger could not dynamically alter the entry-to-element; if you needed to effect the ratio, you had to delete all log streams associated with this structure, and then delete and re-define the structure with the desired values. However, all supported releases of System Logger can now alter this ratio, so it is not as important to code these values. However, as with most functions that heuristic tune themselves, providing a reasonable initial starting point hastens the tuning.

#### ***MAXBUFSIZE(64000)***

Specifies the maximum size of the buffer DFSMStvs can pass to the System Logger when it issues an IXGWRITE. For a CF-based log stream, this value, along with the value specified for AVGBUFSIZE, determine the initial entry-to-element ratio for the structure when it is first allocated. MAXBUFSIZE determines the size of the entries: a value of X or more (I need to

verify this value) results in an entry size of 512, while any value less than X results in an element size of 256. System logger then calculates how many elements are needed to accommodate the average buffer size, as specified in AVGBUFSIZE, and uses that ratio to set the initial entry-to-element ratio.

When System Logger first became available, it was vital to code these two values correctly as System Logger could not dynamically alter the entry-to-element; if you needed to effect the ratio, you had to delete all log streams associated with this structure, and then delete and re-define the structure with the desired values. However, all supported releases of System Logger can now alter this ratio, so it is not as important to code these values. However, as with most functions that heuristic tune themselves, providing a reasonable initial starting point hastens the tuning.

### **Keywords for log stream definitions**

Following is a description for the keywords used to define the log streams in the LOGR policy:

#### ***AUTODELETE and RETPD***

AUTODELETE and RETPD can have a disastrous effect on IGWLOG and IGWSHUNT if specified other than AUTODELETE(NO) and RETPD(0). With AUTODELETE(YES) and RETPD>0, even though DFSMSStvs attempts log tail management, all data is offloaded to the offload data sets and held for the number of days specified for RETPD. AUTODELETE(YES) allows the System Logger (rather than DFSMSStvs) to decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified with RETPD(0), the System Logger deletes the old offload data set (and the data). If DFSMSStvs needs the data for backout, the result is a IGW839I message with a 804 return code and DFSMSStvs terminates.

For forward recovery and the log-of-logs log streams, you might want to have System Logger automatically delete data that is older than three days by coding RETPD(3) and AUTODELETE(YES). However, if you allow System Logger to delete the data, you should ensure that you are taking back-ups of the associated VSAM data sets at a more frequent interval to ensure you have all the data needed to reconstruct the data set. For example, if you only take back-ups every four days, then you should code a retention period of at least 5 (if not 9 to ensure you have two generations of back-ups).

#### ***DIAG(YES)***

DIAG(YES) should always be specified for IGWLOG and IGWSHUNT. In a case where the System Logger is unable to locate data requested by Transactional VSAM, a response of return code 8 with reason code IXGRSNCODENOBLOCK is given. This means backout data is not available and Transactional VSAM treats it as a fatal error, terminating with a IGW839I message. In many cases, information from the System Logger address space is needed to resolve the problem. When DIAG(YES) is specified in the log stream definition, a dump of the System Logger address space is provided (by the System Logger) in addition to the dump provided by Transactional VSAM. There is no overhead associated with this parameter unless a dump is requested. This error is normally seen when Transactional VSAM issues an IXGBRWSE for backout data or during activity keypoint processing when Transactional VSAM issues the IXGDELET command to trim the tail of the log stream. It can also be returned when DFSMSStvs initializes and perform an IXGCONN (connect) to connect to the undo log stream.

#### ***HIGHOFFLOAD***

The HIGHOFFLOAD parameter, in conjunction with the size of the log stream, has a major impact on the amount of virtual storage used by the System Logger. For a Coupling Facility log stream, before data is written to the Coupling Facility, it is written to a buffer in the System

Logger data space. If staging data sets are used, the data is written to the Coupling Facility first, and then written to the staging data set, rather than the data space.

If the HIGHOFFLOAD value is too high, the log stream may fill before offload processing can free up sufficient space in the log stream. Transactional VSAM is not aware the offload process is taking place, and continues to write to the log stream. This can lead to an entry or structure full condition, which causes log writes to be suspended for 3 seconds.

HIGHOFFLOAD should be set at 80 - 85% for all Transactional VSAM log streams.

### **LOGGERDUPLEX**

LOGGERDUPLEX(UNCOND) specifies that System Logger should continue to perform its own log data duplexing even if the structure is exploiting the System-Managed Coupling Facility Structure Duplexing function. LOGGERDUPLEX(COND) allows System Logger to rely on the duplexing of data provided by System-Managed Coupling Facility Structure Duplexing support, which means that System Logger does not need to maintain its own version of duplexed data.

Refer to “Defining CF structures - System Logger policy” on page 32 for a discussion on the values for LOGGERDUPLEX keyword.

Specifying LOGGERDUPLEX(COND) only has any meaning if you are exploiting System-Managed Coupling Facility Structure Duplexing.

### **LOWFFLOAD**

The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the Transactional VSAM environment, the LOWOFFLOAD value should be high enough to retain the data required for backout but low enough to keep the number of offloads to a minimum.

LOWOFFLOAD should be set around 60% for IGWLOG, and 0 for IGWSHUNT, forward recovery log stream and the log-of-logs log stream.

### **LS\_DATACLAS**

With this keyword, you can specify the SMS Dataclass that is used for the allocation of offload data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure the desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the SHAREOPTIONS. Offload data sets may use a CI size up to 24K; in general, the larger CI size provides better performance, however, if the log stream consists mostly of log blocks of a much smaller size, the 24K CI size may be wasteful.

**Important:** The data class is the only way to specify the SHAREOPTIONS for a data set. You **MUST** ensure that all System Logger data sets, including offload data sets, are allocated with a dataclass that defines SHAREOPTIONS (3,3).

### **LS\_SIZE**

LS\_SIZE defines the allocation size for the *offload* data sets. It should be specified large enough to contain several offloads, possibly a day's worth. IGWLOG and IGWSHUNT should only offload a minimal amount of data.

For forward recovery log streams, *all* data is offloaded. It's very important to specify LS\_SIZE large enough to limit the number of times an additional data set is allocated, to reduce the exposure to delays during the allocation process. The size to be specified is based on the amount of data written during the prime processing window of the day. The amount of data

written can be determined using the SMF 88 data produced for the log stream by the System Logger.

If the extent size of the LS\_DATACLAS (or the value specified in LS\_SIZE) is too small, frequent DASD shifts (allocation of a new offload data set) will occur. Frequent DASD shifts have a negative effect on performance and expose the system to a depletion of the offload data sets. The number of offload data sets is limited by the DSEXTENT value specified in the LOGR Couple Data Set. The DSEXTENT value defines the number of logger directory entries. Each directory can point to a maximum of 168 offload data sets. Prior to OS/390 1.3, the number of extents was limited to 1; with OS/390 1.3 and above, the number is limited only by the amount of DASD available.

**Attention:** If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member and the default value in ALLOCxx is 2 tracks. For information about how to change this value, refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592. Using this default value results in many small offload data sets, which adversely effects System Logger performance.

### **OFFLOADRECALL**

OFFLOADRECALL should be set to *NO* for Transactional VSAM log streams. This option was added via APAR OW48404. With OFFLOADRECALL(NO), System Logger does not recall an offload data set that was migrated by HSM, but instead, it allocates a new offload data set. This prevents the System Logger from waiting for the recall and avoids the risk of Transactional VSAM being unable to write to a log stream which has filled while waiting for the offload to proceed.

### **STG\_DUPLEX and DUPLEXMODE**

STG\_DUPLEX and DUPLEXMODE determine under what conditions, if any, System Logger uses a staging data set for a given log stream on a given system.

STG\_DUPLEX(NO) indicates that System Logger never uses staging data sets, regardless of the nature of the connection to the structure on which the log stream resides. Since Transactional VSAM cannot tolerate a loss of data condition on the IGWLOG and IGWSHUNT log streams, you should not code STG\_DUPLEX(NO) for these log streams. (The one exception to this recommendation is in a test environment where you are willing to perform a Transactional VSAM cold start after a double failure.)

STG\_DUPLEX(YES) with DUPLEXMODE(UNCOND) states that System Logger should always use staging data sets for the log stream, even if the connection to the associated structure is failure independent. Even though this guarantees that there will never be a loss of data condition on the log stream, it comes at a price: every write to the log stream is gated by the speed of the I/O to the staging data set.

STG\_DUPLEX(YES) with DUPLEXMODE(COND) means that System Logger only uses a staging data set if the CF on which the structure resides becomes volatile, or the structure resides in the same failure domain as the System Logger system. If none of these conditions exists, System Logger duplexes the log blocks in the System Logger data space. With this setting, you only pay the penalty of writing to a staging data set in those conditions that might reside in a loss of data condition due to a single failure. This is the recommended setting for all Transactional VSAM log streams

### **STG\_DATACLAS**

With this keyword, you can specify the SMS Dataclass that is used for the allocation of staging data sets. The Automatic Class Selection (ACS) routines can override this value so

you should ensure that desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the SHAREOPTIONS. Staging data sets must use a CI size of 4K.

**Important:** The data class is the only way to specify the SHAREOPTIONS for a data set. You **MUST** ensure that all System Logger data sets, including staging data sets, are allocated with a dataclass that defines SHAREOPTIONS (3,3).

### **STG\_SIZE**

STG\_SIZE defines the size of the staging data set to be allocated if one is going to be used. (See the description “STG\_DUPLEX and DUPLEXMODE” on page 99 for details on when a staging data set is used.)

The size of the staging data set (STG\_SIZE) must be large enough to hold as much data as the log stream storage in the Coupling Facility.

Data is written to the staging data set in 4096 byte increments, regardless of the buffer size.

For a Coupling Facility log stream, if you don't code STG\_SIZE or specify STG\_SIZE(0), then System Logger allocates a staging data set as large as the structure to which this log stream is defined; this ensures that the staging data set is not too small compared to the size of the structure. Also, if you increase the size of the structure, the next time the staging data set is allocated, System Logger allocates a larger staging data set.

On the other hand, sometime this might waste DASD storage space since logger doesn't know how many connectors will share the log stream, it always allocate a staging data set as big as the coupling facility structure size and this might be too much. For this reason, it is suggested that you monitor the staging data set configuration through the IXGRPT1 report.

### **STRUCTNAME**

This keyword defines the structure, which must be already defined in the System Logger Policy and which must exist in the active CFRM policy before a connection to this log stream is made, in which the data for this log stream resides.

Multiple log streams can use the same structure, or you can dedicate a given structure to a single log stream.

**Note:** When grouping log streams onto the same structure, you should ensure that all the log streams have similar characteristics. You want the arrival rate of the data along with the average log block size and the maximum buffer size to be the same. You should NOT place IGWLOG and IGWSHUNT log streams in the same structure since those two log streams have such different characteristics. However, it may be appropriate to group several IGWLOG log streams into one structure.

### **Defining DASD-only log streams**

Example 3-8 contains a sample job that defines DASD-only log streams for IGWLOG and IGWSHUNT. Since the forward recovery and log-of-logs log streams are really multi-system in nature, they are not capable of being defined as DASD-only.

*Example 3-8 Defining DASD-only log streams in the System Logger Policy*

```
//DASDONLY JOB CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID.  
//LOGRPOL EXEC PGM=IXCMIAPU  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
```

```

DATA TYPE(LOGR) REPORT(NO)
DEFINE LOGSTREAM NAME(IGWTV001.IGWLOG.SYSLOG)
  AUTODELETE(NO)
  RETPD(0)
  DASDONLY(YES)
  DIAG(YES)
  HIGHOFFLOAD(80)
  LOWOFFLOAD(60)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(2000)
  MAXBUFSIZE(64000)
  OFFLOADRECALL(NO)
  STG_DATACLAS(LOGR4K)
  STG_SIZE(0)
LIST LOGSTREAM NAME(IGWTV001.IGWLOG.SYSLOG) DETAIL(YES)

DEFINE LOGSTREAM NAME(IGWTV001.IGWSHUNT.SHUNTLOG)
  AUTODELETE(NO)
  RETPD(0)
  DASDONLY(YES)
  DIAG(YES)
  HIGHOFFLOAD(80)
  LOWOFFLOAD(0)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(2000)
  MAXBUFSIZE(64000)
  OFFLOADRECALL(NO)
  STG_DATACLAS(LOGR4K)
  STG_SIZE(0)
LIST LOGSTREAM NAME(IGWTV001.IGWSHUNT.SHUNTLOG) DETAIL(YES)

```

/\*

---

## Keywords for log stream definitions

Following is a description of the keywords used to define the log stream in the LOGR policy:

### ***AUTODELETE and RETPD***

AUTODELETE and RETPD can have a disastrous effect on IGWLOG and IGWSHUNT if specified other than AUTODELETE(NO) and RETPD(0). With AUTODELETE(YES) and RETPD>0, even though DFSMSStvs attempts log tail management, all data is offloaded to the offload data sets and held for the number of days specified for RETPD. AUTODELETE(YES) allows the System Logger (rather than DFSMSStvs) to decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified with RETPD(0), the System Logger deletes the old offload data set (and the data). If DFSMSStvs needs the data for backout, the result is a IGW839I message with a 804 return code and DFSMSStvs terminates.

### ***DASDONLY(YES)***

This is how you identify a log stream as DASD-only. You cannot code STG\_DUPLEX, DUPLEXMODE or LOGGERDUPLEX; you can only code this keyword.

### ***DIAG(YES)***

DIAG(YES) should always be specified for IGWLOG and IGWSHUNT. In a case where the System Logger is unable to locate data requested by Transactional VSAM, a response of return code 8 with reason code IXGRSNCODENOBLOCK is given. This means backout data is not available and Transactional VSAM treats it as a fatal error, terminating with a IGW839I

message. In many cases, information from the System Logger address space is needed to resolve the problem. When DIAG(YES) is specified in the log stream definition, a dump of the System Logger address space is provided (by the System Logger) in addition to the dump provided by Transactional VSAM. There is no overhead associated with this parameter unless a dump is requested. This error is normally seen when Transactional VSAM issues an IXGBRWSE for backout data or during activity keypoint processing when Transactional VSAM issues the IXGDELET command to trim the tail of the log stream. It can also be returned when DFSMStvs initializes and issues an IXGCONN (connect) to connect to the log stream.

### **HIGHOFFLOAD**

The HIGHOFFLOAD parameter, in conjunction with the size of the log stream, has a major impact on the amount of virtual storage used by the System Logger. For a Coupling Facility log stream, before data is written to the Coupling Facility, it is written to a buffer in the System Logger data space. If staging data sets are used, the data is written to the Coupling Facility first, and then written to the staging data set, rather than the data space.

If the HIGHOFFLOAD value is too high, the log stream may fill before offload processing can free up sufficient space in the log stream. Transactional VSAM is not aware the offload process is taking place, and continues to write to the log stream. This can lead to an entry or structure full condition, which causes log writes to be suspended for 3 seconds.

HIGHOFFLOAD should be set at 80–85% for all Transactional VSAM log streams.

### **LOWFFLOAD**

The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the Transactional VSAM environment, the LOWOFFLOAD value should be high enough to retain the data required for backout but low enough to keep the number of offloads to a minimum.

LOWOFFLOAD should be set around 60% for IGWLOG, and 0 for IGWSHUNT.

### **LS\_DATACLAS**

With this keyword, you can specify the SMS Dataclass that is used for the allocation of offload data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure the desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the SHAREOPTIONS. Offload data sets may use a CI size up to 24 K. In general, the larger CI size provides better performance, however, if the log stream consists mostly of log blocks of a much smaller size, the 24 K CI size may be wasteful.

**Important:** The data class is the only way to specify the SHAREOPTIONS for a data set. You *must* ensure that all System Logger data sets, including offload data sets, are allocated with a dataclass that defines SHAREOPTIONS (3,3).

### **LS\_SIZE**

LS\_SIZE defines the allocation size for the *offload* data sets. It should be specified large enough to contain several offloads, possibly a day's worth. IGWLOG and IGWSHUNT should only offload a minimal amount of data.

If the extent size of the LS\_DATACLAS (or the value specified in LS\_SIZE) is too small, frequent DASD shifts (allocation of a new offload data set) will occur. Frequent DASD shifts have a negative effect on performance and expose the system to a depletion of the offload data sets. The number of offload data sets is limited by the DSEXTENT value specified in the



LOGR Couple Data Set. The DSEXTENT value defines the number of logger directory entries. Each directory can point to a maximum of 168 offload data sets. Prior to OS/390 1.3, the number of extents was limited to 1; with OS/390 1.3 and above, the number is limited only by the amount of DASD available.

**Attention:** If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member and the default value in ALLOCxx is two tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592. Using this default value results in many small offload data sets, which adversely effects System Logger performance.

### **OFFLOADRECALL**

OFFLOADRECALL should be set to *NO* for Transactional VSAM log streams. This option was added via APAR OW48404. With OFFLOADRECALL(NO), System Logger does not recall an offload data set that was migrated by HSM, but instead, it allocates a new offload data set. This prevents the System Logger from waiting for the recall and avoids the risk of Transactional VSAM being unable to write to a log stream which has filled while waiting for the offload to proceed.

### **STG\_DATACLAS**

With this keyword, you can specify the SMS Dataclass that is used for the allocation of staging data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure that desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the SHAREOPTIONS. Staging data sets must use a CI size of 4 K.

**Important:** The data class is the only way to specify the SHAREOPTIONS for a data set. You *must* ensure that all System Logger data sets, including staging data sets, are allocated with a dataclass that defines SHAREOPTIONS (3,3).

### **STG\_SIZE**

STG\_SIZE defines the size of the staging data set to be allocated. For DASD-only log streams, staging data sets are always used.

Data is written to the staging data set in 4096 byte increments, regardless the buffer size.

The rule of thumb is it must be large enough to hold the data written during an activity keypoint interval, plus the length of time of the longest running unit of work.

**Attention:** If STG\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by STG\_DATACLAS, the value is taken from the ALLOCxx Parmlib member and the default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

A staging data set with a size of only 2 tracks has an adverse effect on the performance of a DASD-only log stream.

You should monitor the SMF 88 records (using either the sample program IXGRPT1 or IXGRPT1J to format that data) to ensure that the staging data sets are properly sized.

## 3.2.4 Security definitions

DFSMStvs requires read and write access to all the log streams that it uses and you have to ensure that the proper definitions are made to the security product (like RACF®) to allow this access while at the same time prohibiting undesired accesses.

The easiest way to grant DFSMStvs access to its log streams is to assign the RACF attribute of PRIVILEGED or TRUSTED to the VSAM RLS server address space, which is called SMSVSAM. With either of these attributes, most RACROUTE REQUEST=AUTH requests are deemed successful without performing any checks and the SMSVSAM address space can access its log streams without any additional profiles.

If SMSVSAM is not given the PRIVILEGED or TRUSTED attribute, then the user ID associated with the SMSVSAM address space must have UPDATE access to all the log streams that DFSMStvs uses.

All log streams are protected by profiles in the LOGSTRM class, and if you use a good naming convention for all your log streams, you can protect the log streams with a few generic profiles. For example, the undo log streams start with a given pattern of IGWTV&tvname (where &tvname is defined in IGDSMSxx and is a number in the range of 001 to 999). First you want to ensure that no unexpected programs can access the log streams by defining a profile in the LOGSTRM class with universal access of NONE. For an instance of DFSMStvs with a suffix of 001 that is assigned a user ID of smsvsam\_userid, the definition of the profile looks like that shown in Example 3-9.

*Example 3-9 Defining the generic profile to protect the undo log streams*

---

```
RDEFINE LOGSTRM IGWTV001.** UACC(NONE)
```

---

Now you must grant the SMSVSAM address space UPDATE access to that profile, as shown in. Example 3-10

*Example 3-10 Granting access to the undo log streams to the DFSMStvs user ID*

---

```
PERMIT IGWTV001.** CLASS(LOGSTRM) ACCESS(UPDATE) USERID(smsvsam_userid)
```

---

The forward-recovery log streams and the log-of-log log stream must be protected in a similar fashion. However, in addition to the user ID associated with the SMSVSAM address space, the user ID of all the forward recovery products need at least READ access (and UPDATE if they are to perform the log-tail trimming for the log streams). If all the forward recovery log streams start with the same prefix of FORWARD, the name of the log-of-logs log stream is LOG.OF.LOGS and the user ID associated with the forward recovery program (like CICSVR) is rebuild\_userid, the definition of the profiles and the PERMITs might look like that shown in Example 3-11.

*Example 3-11 Profiles for protecting forward recovery and log-of-log log streams*

---

```
RDEFINE LOGSTRM FORWARD.** UACC(NONE)
RDEFINE LOGSTRM LOG.OF.LOGS UACC(NONE)
PERMIT FORWARD.** CLASS(LOGSTRM) ACCESS(UPDATE) USERID(smsvsam_userid)
PERMIT FORWARD.** CLASS(LOGSTRM) ACCESS(UPDATE) USERID(rebuild_userid)
PERMIT LOG.OF.LOGS CLASS(LOGSTRM) ACCESS(UPDATE) USERID(smsvsam_userid)
PERMIT LOG.OF.LOGS CLASS(LOGSTRM) ACCESS(UPDATE) USERID(rebuild_userid)
```

---

To perform peer-recovery, one Transactional VSAM instance might have to access the log streams from another instance of Transactional VSAM. To allow this access with the fewest number of security profiles, we recommend that you associate the same user ID with each instance of the SMSVSAM address space.

## 3.3 Operational considerations

The following are items to consider when you are using TVS:

### 3.3.1 Can you stop/start the use of Logger

System Logger provides a critical function within Transactional VSAM, and Transactional VSAM cannot operate without the services of the System Logger. If System Logger is not available when Transactional VSAM initializes, it does not complete initialization and waits for System Logger to become available. Unfortunately, there is no message issued to indicate that Transactional VSAM is waiting for System Logger.

If System Logger should fail after Transactional VSAM initializes, Transactional VSAM does not detect the loss of System Logger until the next request for Transactional VSAM services comes from a batch job. At that point, Transactional VSAM issues the following error messages (Example 3-12).

*Example 3-12 Messages from DFSMStvs when it detects System Logger is gone*

---

```
IGW839I 05012003 08.15.09 AN ERROR OCCURRED DURING SYSTEM LOGGER
OPERATION IXGWRITE FOR SYSTEM LOG STREAM
IGWTV002.IGWLOG.SYSLOG.
SYSTEM LOGGER RETURN CODE 00000008 REASON CODE 00000890
```

```
IGW827I 05012003 08.15.09 A FAILURE HAS OCCURRED WHILE WRITING TO
THE SYSTEM LOG IGWTV002.IGWLOG.SYSLOG . ACCESS TO THE SYSTEM LOG
HAS BEEN LOST. TRANSACTIONAL VSAM WILL BE TERMINATED.
```

```
IGW471I DFSMS VSAM RLS REQUEST TO DISABLE
TRANSACTIONAL VSAM INSTANCE IGWTV002 IS ACCEPTED.
DISABLE REASON:
TRANSACTIONAL VSAM DETECTED SYSTEM LOGGER ENVIRONMENTAL ERROR
```

---

After this, Transactional VSAM is unavailable.

### 3.3.2 Defining log streams dynamically

Unlike CICS TS, Transactional VSAM does not support log stream models; every log stream that Transactional VSAM needs must be defined in the System Logger policy BEFORE Transactional VSAM attempts to use it. How Transactional VSAM reacts to a missing log stream depends on which log stream is not defined in the LOGR inventory:

Undefined log stream	How Transactional VSAM reacts
Undo log stream	Transactional VSAM does not initialize
forward recovery log stream	Transactional VSAM fails the open of the data set
log-of-logs	Transactional VSAM continues processing without it

### 3.3.3 Displaying log stream information

In addition to the System Logger DISPLAY LOGGER command (refer to 7.1.1, “Display Logger command” on page 254), Transactional VSAM displays log stream information in response to several commands. The D SMS,TRANVSAM command displays information about Transactional VSAM in general, including log stream information. Example 3-13 is an example of the command response.

*Example 3-13 Output from D SMS,TRANVSAM command*

---

```

IEE932I 033
IGW800I 08.42.44 DISPLAY SMS,TRANSACTIONAL VSAM

DISPLAY SMS,TRANSACTIONAL VSAM - SERVER STATUS
System   TVSNAME  State  Rrs   #Urs   Start   AKP   QtimeOut
-----
#@$2    IGWTV002 ACTIVE REG           0 WARM/WARM   5000   300

DISPLAY SMS,TRANSACTIONAL VSAM - LOGSTREAM STATUS
LogStreamName      State   Type      Connect Status
-----
IGWTV002.IGWLOG.SYSLOG   Enabled  UnDoLog   Connected
IGWTV002.IGWSHUNT.SHUNTLOG Enabled  ShuntLog  Connected
#@$C.CICSVR.LGOFLOGS    Enabled  LogOfLogs Connected
#@$#.CICSVR.F01DACCT    Enabled  FrLog     DisConnected

```

---

You can also get more detailed information about a single log stream (or all log streams) using the D SMS,LOG command. You can either specify the name of a specific log stream or use the keyword ALL, which results in information for ALL log streams known to all instances of Transactional VSAM within the sysplex. Example 3-14 is the output from the display request against a single log stream using D SMS,LOG(IGWTV002.IGWLOG.SYSLOG).

*Example 3-14 Output from D SMS,LOG(IGWTV002.IGWLOG.SYSLOG) command*

---

```

IGW804I 08.51.32 DISPLAY SMS,LOG

DISPLAY SMS,LOG - LOG STREAM STATUS
Name: IGWTV002.IGWLOG.SYSLOG   State: Enabled   Type: UnDo

System   TVSNAME  JobName  Urid of Oldest Log Block
-----
#@$2    IGWTV002 TVSV07C  B95AAE067E743374000005FD01030000*

DISPLAY SMS,LOG - LOG STREAM USAGE
LogStreamName: IGWTV002.IGWLOG.SYSLOG
System   TVSNAME  JobName  JobName  JobName  JobName  JobName
-----
#@$2    IGWTV002 TVSV05C  TVSV07C  TVSV08C  TVS006C  TVS007C

#@$2    IGWTV002 TVS009C

*OLDEST URID ACROSS ALL SYSTEMS IN THE SYSPLEX

```

---

Since Transactional VSAM cannot delete any log blocks that contain data associated with the oldest unit of recovery, you can use this command to determine which UR from which batch job is preventing Transactional VSAM from pruning the tail of the undo log stream.

### 3.3.4 Monitoring for offload conditions

The only way to monitor how many offload events occur within a given interval is to gather the SMF 88 records and use either the sample program IXGRPT1 or IXGRPT1J to format the data.

Unfortunately, short of taking a dump of the IXGLOGR address space, there is no way to determine how long a given offload event takes to complete.

## 3.4 Recovery

In this section, we discuss how Transactional VSAM reacts to failures that might happen in the logger environment.

### 3.4.1 Missing data in the log streams

Transactional VSAM reacts to missing data in a different log stream based on the type of log stream that is affected.

First of all, Transactional VSAM only recognizes that there are missing data on the IGWLOG and IGWSHUNT log stream because these are the only log streams where TVS retrieves data from. If any data is lost on the log-of-logs or in any of the forward-recovery log, TVS will not detect it. It is CICSVR or similar that sees the missing data condition asynchronously from the time that the problem occurred.

The impact of having missing data on either IGWLOG or IGWSHUNT to a Transactional VSAM subsystem, is that it will not be able to retrieve data to perform any backout requests. When Transactional VSAM recognizes that there missing data, it comes down and requests a cold start. For information about how to cold start Transactional VSAM, refer to *DFSMSStvs Planning and Operating Guide*, SC26-7348-00. Example 3-15 is a syslog sample of the scenario.

*Example 3-15 Missing data in IGWLOG log stream*

---

```
IGW839I 07282003 14.03.32 AN ERROR OCCURRED DURING SYSTEM LOGGER
OPERATION IXGBRWSE READBLOCK FOR SYSTEM LOG STREAM
IGWTV002.IGWLOG.SYSLOG.
SYSTEM LOGGER RETURN CODE 00000008 REASON CODE 00000804
```

```
IGW822I 07282003 14.03.32 THE SYSTEM LOGGER FAILED TO LOCATE A BLOCKID
REQUESTED BY TRANSACTIONAL VSAM. MISSING BLOCKID 0000000133C3B4C8
CHAIN HISTORY POINT 0000000133BF5F00
```

```
IGW833I A FAILURE OCCURRED WHILE READING FROM SYSTEM LOG
IGWTV002.IGWLOG.SYSLOG . THE REQUESTED DATA COULD NOT BE FOUND.
TRANSACTIONAL VSAM WILL BE QUIESCED SO INFLIGHT TASKS CAN
COMPLETE. NEXT TRANSACTIONAL VSAM START MUST BE COLD
```

```
IGW835I A FAILURE TO READ DATA FROM SYSTEM LOG
IGWTV002.IGWLOG.SYSLOG DURING BACKOUT HAS CAUSED JOB **AKP**
STEP **AKP** UNIT OF RECOVERY B9C98A7B7E7491440000000701020000
TO BE SHUNTED
```

```
IGW473I DFSMS VSAM RLS REQUEST TO QUIESCE
TRANSACTIONAL VSAM LOGSTREAM IGWTV002.IGWLOG.SYSLOG IS ACCEPTED
QUIESCE REASON: TRANSACTIONAL VSAM LOGGER DETECTED AN I/O ERROR
*** TRANSACTIONAL VSAM COLD START IS REQUIRED ***
```

```
IGW834I TRANSACTIONAL VSAM RECEIVED A LOST DATA WARNING FOR
SYSTEM LOG IGWTV002.IGWLOG.SYSLOG.
TRANSACTIONAL VSAM WILL BE QUIESCED SO INFLIGHT TASKS
CAN COMPLETE. NEXT TRANSACTIONAL VSAM START MUST BE COLD
```

```
IGW10114I TVS003C G003 B9C98A7B7E7491440000000701020000 4
FAILURE OCCURRED DURING BACKOUT PROCESSING WHILE
TRYING TO START A BROWSE OF THE UNDO LOG.
LOGGER RETURN CODE (00000008)
LOGGER REASON CODE (7003000A)
```

```
ATR306I RESOURCE MANAGER IGWTV002020031960915105983550000
CAUSED A HEURISTIC-MIXED CONDITION FOR URID =
B9C98A7B7E7491440000000701020000.
```

---

The impact of having missing data in the log-of-log will not be so dramatic. It only affects performance during the CICSVR operations. DFSMSStvs continues to function without this log stream.

For a forward-recovery log stream, a missing data condition prevents the possibility to reconstruct the file by applying these change to the image copy. DFSMSStvs closes the data set. To recover the condition, you need to close all the data sets associated with that log stream, delete the log stream, take a data sets backup, define the log stream and e-open the data sets.

### 3.4.2 Log stream full condition

There are two types of full conditions: one is temporary and it is when the interim storage gets filled up and the second one is permanent when the complete log stream (interim plus offloads) is full.

In the first case, you will see message IGW838I to inform you of the delay but this situation should also resolve as soon as data gets offloaded to DASD. If you see this message very often, you should re-evaluate your logger configuration and ensure it is tuned properly.

Example 3-16 is a sample of the syslog message for this scenario.

*Example 3-16 Interim storage full condition*

---

```
IGW838I 07282003 15.32.14 A TEMPORARY ERROR CONDITION OCCURRED DURING
SYSTEM LOGGER OPERATION IXGWRITE FOR LOG STREAM
IGWTV002.IGWLOG.SYSLOG . SYSTEM LOGGER RETURN CODE 00000008
REASON CODE 00000860
```

---

The permanent condition, where the log stream is completely full, can happen in the following situations:

- ▶ When there is no more space on the DASD pool to allocate space for the next offload data set
- ▶ When the DSEXTENT has been saturated

If this situation happens against either the IGWLOG or IGWSHUNT log stream, Transactional VSAM cannot continue and it will come down. The following are the syslog messages generated in this scenario. While Transactional VSAM is down, system logger will keep an internal connection to the log stream to preserve the copy of the data that cannot be

offloaded. You can resolve the problem either by adding more space to the offload data set DASD pool or by switching to new LOGR data set formatted with a bigger DSEXTENT value. Once you have resolved the problem, you need to restart Transactional VSAM. At restart time, Transactional VSAM reconnects to the log stream and the offload process resumes. See Example 3-17.

*Example 3-17 Log stream full condition*

---

```
IXG251I IKJ56245I DATA SET IXGLOGR.IGWTV002.IGWLOG.SYSLOG.A0000582
NOT ALLOCATED, NOT ENOUGH SPACE ON VOLUMES+
IXG251I IKJ56245I USE DELETE COMMAND TO DELETE UNUSED DATA SETS
```

```
IXG301I SYSTEM LOGGER FAILED TO OFFLOAD DATA FOR LOG STREAM
IGWTV002.IGWLOG.SYSLOG IN STRUCTURE LOG_IGWLOG_001. RETURN CODE:
00000008 REASON CODE: 00000805 DIAG1: 00000004 DIAG2: 4714041D
DIAG3: 0107001B DIAG4: 00000000
```

```
IGW839I 07282003 15.56.37 AN ERROR OCCURRED DURING SYSTEM LOGGER
OPERATION IXGWRITE FOR SYSTEM LOG STREAM
IGWTV002.IGWLOG.SYSLOG.
SYSTEM LOGGER RETURN CODE 00000008 REASON CODE 0000085D
```

```
IGW827I 07282003 15.56.37 A FAILURE HAS OCCURRED WHILE WRITING TO
THE SYSTEM LOG IGWTV002.IGWLOG.SYSLOG . ACCESS TO THE SYSTEM LOG
HAS BEEN LOST. TRANSACTIONAL VSAM WILL BE TERMINATED.
```

```
IGW471I DFSMS VSAM RLS REQUEST TO DISABLE
TRANSACTIONAL VSAM INSTANCE IGWTV002 IS ACCEPTED.
DISABLE REASON:
TRANSACTIONAL VSAM DETECTED SYSTEM LOGGER ENVIRONMENTAL ERROR
```

```
IGW1011I TVSD17C GD17 B9C9A3C77E7480000000D1FB01020000
FAILURE OCCURRED DURING PREPARE PROCESSING
WHILE TRYING TO WRITE A COMMIT RECORD TO THE UNDO LOG.
LOGGER RETURN CODE (00000008)
LOGGER REASON CODE (7002001F)
THIS UR WILL BE SHUNTED
```

```
ATR306I RESOURCE MANAGER IGWTV002020031960915105983550000
CAUSED A HEURISTIC-MIXED CONDITION FOR URID =
B9C9A3C77E7480000000D1FB01020000.
```

---

If this situation happens against a forward-recovery log stream, Transactional VSAM closes the data set(s) associated with the log stream since it cannot process the log data.

If this happens against the log-of-logs log stream, Transactional VSAM disables the use of the log stream but it continues to function.

### 3.4.3 CF failures

In case of a coupling facility failure or a connectivity to a coupling facility failure, system logger detects the loss of connectivity to the single instance of the structure. Then the system that recognized the failure condition initiates a rebuild for the structure, according to the PREFLIST specified in the structure definition for the CFRM policy.

While the recovery is taking place, Transactional VSAM will not be able to access the log streams being rebuilt and receives return code 861 temporary unavailable from system logger. This situation should resolve as soon as the rebuild completes and the log stream is

available in the new structure. This temporary error condition will be handled by Transactional VSAM and not propagated to the applications. Example 3-18 is a sample of the log messages expected during this scenario.

*Example 3-18 Log stream rebuild log messages*

---

```
IGW838I 07282003 16.48.22 A TEMPORARY ERROR CONDITION OCCURRED DURING
SYSTEM LOGGER OPERATION IXGWRITE FOR LOG STREAM
IGWTV002.IGWLOG.SYSLOG . SYSTEM LOGGER RETURN CODE 00000008
REASON CODE 00000861
```

---

### 3.4.4 System Logger failure

As a result of the system logger address space going down, Transactional VSAM reacts in the following ways:

- ▶ Since it is not be able to access the IGWLOG and IGWSHUNT, the instance of Transactional VSAM running on the system where system logger went down will disconnect from the log streams and then terminate
- ▶ It will disconnect from the other log streams, forward recovery and log-of-logs, but these log streams are still available on the other systems in the sysplex.

When the system logger address space, Transactional VSAM automatically restarts and reconnects to the log streams. Example 3-19 is a syslog of the scenario.

*Example 3-19 System Logger failure messages*

---

```
IGW473I DFSMS VSAM RLS REQUEST TO MAKE
TRANSACTIONAL VSAM LOGSTREAM #@$C.CICSVR.LGOFLOGS
UNAVAILABLE FOR USE ON THIS SYSTEM IS ACCEPTED.
TRANSACTIONAL VSAM LOGSTREAM #@$C.CICSVR.LGOFLOGS IS MARKED
UNAVAILABLE ON SYSTEM: #@$2
UNAVAILABLE REASON: TRANSACTIONAL VSAM DETECTED A LOCAL LOG ERROR
IGW855I TRANSACTIONAL VSAM FAILED TO WRITE TO
THE LOG OF LOGS
#@$C.CICSVR.LGOFLOGS , WHILE PROCESSING DSN
TVSV.VF01D.TRANLOG , FORWARD RECOVERY
LOG STREAM #@$C.CICSVR.LOG001.
RETURN CODE 00000008 REASON CODE 70060016
```

```
IGW839I 07282003 16.38.02 AN ERROR OCCURRED DURING SYSTEM LOGGER
OPERATION IXGWRITE FOR SYSTEM LOG STREAM
#@$C.CICSVR.LOG001.
SYSTEM LOGGER RETURN CODE 00000008 REASON CODE 00000890
```

```
IGW839I 07282003 16.38.02 AN ERROR OCCURRED DURING SYSTEM LOGGER 4
OPERATION IXGWRITE FOR SYSTEM LOG STREAM
IGWTV002.IGWLOG.SYSLOG.
SYSTEM LOGGER RETURN CODE 00000008 REASON CODE 00000890
```

```
IGW827I 07282003 16.38.03 A FAILURE HAS OCCURRED WHILE WRITING TO
THE SYSTEM LOG IGWTV002.IGWLOG.SYSLOG . ACCESS TO THE SYSTEM LOG
HAS BEEN LOST. TRANSACTIONAL VSAM WILL BE TERMINATED.
```

```
IEF402I IXGLOGR FAILED IN ADDRESS SPACE 001F
SYSTEM ABEND S1C5 - REASON CODE 020004
IXG058E LOGSTREAM CONNECTIONS HAVE BEEN LOST
```



DUE TO SYSTEM LOGGER TERMINATION  
IXG056I SYSTEM LOGGER ADDRESS SPACE HAS ENDED.

---

*Example 3-20 Transactional VSAM status after failure*

---

D SMS,TRANVSAM

```
IGW800I 16.42.45 DISPLAY SMS,TRANSACTIONAL VSAM
DISPLAY SMS,TRANSACTIONAL VSAM - SERVER STATUS
  System  TVSNMAME  State  Rrs  #Urs  Start  AKP  QtimeOut
-----
#@$2     IGWTV002  DISED  UNREG      1  COLD/WARM  1000  300
DISPLAY SMS,TRANSACTIONAL VSAM - LOGSTREAM STATUS
  LogStreamName          State      Type      Connect Status
-----
IGWTV002.IGWLOG.SYSLOG   Disabled  UnDoLog   DisConnected
IGWTV002.IGWSHUNT.SHUNTLOG Disabled  ShuntLog  DisConnected
#@$C.CICSVR.LGOFLOGS     Ena-Unava1 LogOfLogs  DisConnected
#@$C.CICSVR.LOG001       Ena-Unava1 FrLog     DisConnected
```

---

*Example 3-21 System logger restart*

---

S IXGLOGRS

```
IGW474I DFSMS VSAM RLS REQUEST TO ENABLE
TRANSACTIONAL VSAM LOGSTREAM IGWTV002.IGWLOG.SYSLOG IS ACCEPTED
IGW471I DFSMS VSAM RLS COMMAND PROCESSOR 608
ON SYSTEM: #@$2
IS WAITING FOR A RESPONSE
FROM TRANSACTIONAL VSAM: IGWTV002
COMMAND REQUESTED:
ENABLE TRANSACTIONAL VSAM: IGWTV002
IGW473I DFSMS VSAM RLS COMMAND PROCESSOR
ON SYSTEM: #@$2
IS WAITING FOR A RESPONSE FROM TRANSACTIONAL VSAM: IGWTV002
COMMAND REQUESTED: ENABLE LOGSTREAM: IGWTV002.IGWLOG.SYSLOG
IGW473I DFSMS VSAM RLS COMMAND PROCESSOR
ON SYSTEM: #@$2
IS WAITING FOR A RESPONSE FROM TRANSACTIONAL VSAM: IGWTV002
COMMAND REQUESTED: ENABLE LOGSTREAM: IGWTV002.IGWSHUNT.SHUNTLOG
IGW860I TRANSACTIONAL VSAM HAS SUCCESSFULLY REGISTERED WITH RLS
IGW888I TRANSACTIONAL VSAM PERMITNONRLSUPDATE EXIT NOT
LOADED FOR INSTANCE IGWTV002 ON SYSTEM #@$2
IGW848I 07282003 16.44.24 SYSTEM UNDO LOG IGWTV002.IGWLOG.SYSLOG
INITIALIZATION HAS STARTED
IXL014I IXLCONN REQUEST FOR STRUCTURE LOG_IGWLOG_001 629
WAS SUCCESSFUL. JOBNAME: IXGLOGR ASID: 0033
CONNECTOR NAME: IXGLOGR #@$2 CFNAME: FACILOG
IGW474I DFSMS VSAM RLS IS CONNECTING TO
TRANSACTIONAL VSAM LOGSTREAM IGWTV002.IGWLOG.SYSLOG
SYSTEM NAME:          #@$2
TRANSACTIONAL VSAM INSTANCE NAME: IGWTV002
IGW848I 07282003 16.44.26 SYSTEM UNDO LOG IGWTV002.IGWLOG.SYSLOG
INITIALIZATION HAS ENDED
```

---

### 3.4.5 Subsystem failure

When the Transactional VSAM instance fails, all the new request (put/get/open) against the VSAM files receives a return code documenting that Transactional VSAM is not available. The application code needs to handle these return codes. For a complete list of these return codes, refer to *DFSMStvs Administration Guide*, GC26-7483-00.

For the in-flight requests, retained locks are generated against the records that were held by that unit of works. Refer to *DFSMStvs Planning and Operating Guide*, SC26-7348-00, to understand how to resolve a situation with retain locks.

## 3.5 Performance

After your initial log streams allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments are necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70-78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70 -78 records

Transactional VSAM log streams are both funnel and active log streams. For the active log streams, the Undo log set, Transactional VSAM will take care of maintaining currency of data within the log streams. For this reason, after an accurate tuning, we should see that offload are eliminated or minimized for these log streams.

On the other log streams, forward-recovery and log-of-logs, Transactional VSAM does not provide any mechanism for maintaining the currency of the data. For these reasons, on these log streams, a large amount of data is expected and so offloads are expected as part of a normal process.

There are some fields in the SMF88 that you might want to check against the active log streams to determine if they are tuned properly: DASD SHIFT, STRUCTURE FULL, ENTRY FULL and OFFLOAD.

As regard to the funnel type log streams, you should check the fields DASD SHIFT, STRUCTURE FULL, ENTRY FULL and SC1/2/3 in the SMF88 for this particular log stream.

Refer to the 8.6.3, "SMF Type 88 records and IXGRPT1 program" on page 291 for a description of these fields and what to do in case one of these conditions happens in your installation.

If you are running Transactional VSAM with CICS, there is no effect on CICS logging for DFHLOG and DFHSHUNT. There should be minimal impact for the log-of-logs and for the forward recovery logging since they are funnel-type log streams. Mainly, you should ensure that there is enough space on the DASD pool for the offload data sets, since the volume of data is probably going to increase.

# IMS Common Queue Server and the System Logger

This chapter describes how the IMS shared queues environment exploits the System Logger to ensure the recoverability of the shared message queues in the event of a loss of the shared queue structures. Specific topics include:

- ▶ Overview of the IMS shared queues environment, including the *Common Queue Server* (CQS) and the System Logger
- ▶ How CQS uses the System Logger
- ▶ How to define the System Logger log streams to the IMSplex
- ▶ CQS and System Logger operational considerations
- ▶ CQS and System Logger error recovery considerations
- ▶ CQS and System Logger performance considerations

Within this chapter, we will use the term *IMSplex* to refer to one or more IMS systems and associated address spaces running in a Parallel Sysplex with data sharing and shared queues. The term *shared queues group* refers to those IMS control regions that are sharing the message queues.

## 4.1 Functional description

This section provides an overview of the shared queues components of an IMSplex environment.

### 4.1.1 Overview of shared queues in the IMSplex

The term *IMSplex* is used by IMS to describe an IMS environment invoking Parallel Sysplex services for sharing data, message queues, and other resources. When used to share the message queues, IMS uses the *Common Queue Server (CQS)* to put messages on, and retrieve messages from, shared queue structures on a *Coupling Facility*. CQS uses the System Logger to log this activity. These log records are required only for shared queue structure recovery. Each IMS system continues to log its own information in the *online log data sets (OLDS)* unique to each IMS.

Figure 4-1 shows a high level view of the IMS components required for two IMSs to share a common set of message queues.

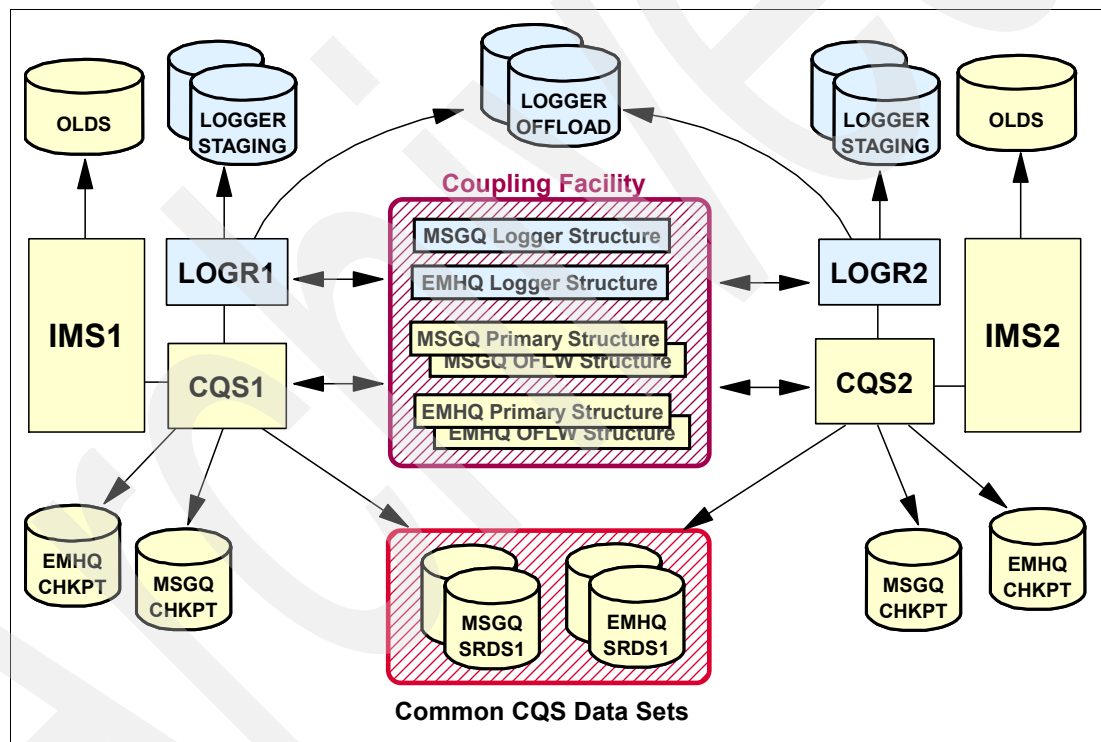


Figure 4-1 IMSplex shared queues component architecture

#### IMS subsystems and the OLDS

Each IMS subsystem (IMS1 and IMS2 in Figure 4-1) continues to provide connectivity to its logged-on end users. In a non-shared queues environment, IMS logs input messages received from these end users in its OLDS and queues the messages on a local transaction queue in its *message queue pool (QPOOL)*. Since it resides only locally, only that IMS has access to the message (transaction). When the transaction is scheduled (locally) and a response returned, IMS logs the response message on the OLDS and queues it in the same message queue pool. These log records can be used to recover the message queues should an error occur resulting in the loss of the queues.

To share these message queues across multiple IMSs in the IMSplex, IMS execution parameters are set up to define a *shared queues group* to which each IMS belongs. In a shared queues environment, IMS still logs the input message on the OLDS as before, but instead of queuing it in a local message queue pool, IMS issues a PUT request to CQS to queue the message on a transaction queue in a Coupling Facility list structure. Once CQS has accepted the message, IMS deletes it from its local queue pool. Note, however, that these IMS log records cannot be used to recover the shared queues.

## Common Queue Server

The Common Queue Server (CQS) functions as a server between IMS and the shared queue structures. IMS does not connect directly to these structures, but uses CQS services to put messages on the shared queue, to read messages from the shared queue, to move a message from one queue to another, and to delete messages when processing is complete. CQS is required on each z/OS image where an IMS control region sharing the message queues is running.

*CQS is responsible for the integrity of these queues - not IMS.* When you are not using shared queues, IMS recovers local message queues by using a copy of the message queues on the IMS logs (for example, a SNAPQ or DUMPQ) and then applying later-logged changes to that copy. When you are using shared queues, CQS follows an equivalent process. A copy of the contents of the shared queues structure is written periodically to a data set called the *Structure Recovery Data Set (SRDS)*. Changes to the shared queues are logged by CQS. However, CQS does not have its own internal logger like IMS does. Instead, CQS uses the System Logger to log updates to the shared queues. Until CQS has successfully logged these changes, no updates are made to the shared queue structure.

### Structure recovery data sets (SRDSs)

There is one set of SRDSs for the entire shared queues group. They are shown in Figure 4-1 on page 114 as SRDS1 and SRDS2. These establish a base point from which the log records can be applied to recover the shared queues if they are lost. The process of making the copy is called a *structure checkpoint* and is usually initiated by an IMS command, as follows:

```
/CQCHKPT SHAREDQ STRUCTURE structure-name
```

There are two SRDSs for full function and two for Fast Path EMH. This provides CQS with a way to recover the structures even if the most recent SRDS is unusable for some reason. However, this also means that CQS must keep all the log records back to the oldest structure checkpoint.

Each structure checkpoint uses the older of SRDS1 and SRDS2. When it completes successfully, CQS issues a call to the System Logger to delete those log records which are older than the older of the two SRDSs. By taking frequent structure checkpoints, you can limit the amount of log data kept by the System Logger in the log stream. We'll address this in more detail later in this chapter.

Note that, no matter which CQS invokes structure checkpoint, the same set of SRDSs are used. A method of serialization is provided to prevent two CQSs from taking a structure checkpoint at the same time.

### CQS checkpoint data sets

Each CQS also takes a *system checkpoint* periodically. CQS system checkpoint data is written to the log stream and the log token associated with that log record is written to the checkpoint data set and to the shared queue structure. The checkpoint data is used when CQS is restarted to resynchronize CQS and IMS. Note, however, that this system checkpoint data may be deleted from the log stream if it is older than the older of the two SRDSs such as could happen if a particular CQS is down. If this happens, the next CQS restart will be a cold

start. There is, however, no loss of message queue integrity, and no loss of messages, unless IMS is also cold started (COLDCOMM) since IMS has enough information on its logs to resynchronize the queues.

## Shared queue structures

IMSS in a shared queues group can share a common set of messages by putting them in a shared queue list structure in a Coupling Facility. You will note in Figure 4-1 on page 114 that there are two of everything - one labeled MSGQ and the other EMHQ. These represent messages queued by IMS as *full function* messages, or fast path *Expedited Message Handler* (EMH) messages. EMH messages are defined to IMS separately and treated differently than full function messages. They have their own set of local and shared message queue structures, and CQS uses a separate log stream to log changes to these queue structures. The rest of this section, and this chapter, addresses *full function message queues* (MSGQ) only, but be aware that everything said also applies to the *fast path EMH message queues* (EMHQ).

### Primary structure

This is the only shared queue structure that is always required in a shared queues group. It is implemented as a list structure and is the primary repository for IMS full function messages (and some control information). Within this structure there are list headers for input transactions, output messages, remote MSC destinations, and several special purpose list headers which are not of interest here.

In normal operation, there typically will be very few messages in the primary structure. Messages only reside in the structure for the life of the transaction. As soon as the transaction completes, the messages will be deleted.

### Overflow structure

This list structure is optional. If it is defined, it is used to prevent the primary structure from filling up. Individual queues (for example, a queue of messages to a printer which is out of paper) may get long and utilize a lot of space in the structure. These queues are identified by CQS and moved from the primary structure to the overflow structure, reclaiming space in the primary structure and allowing normal processing to continue. Any more messages for those queues are put directly on the queues in the overflow structure instead of the primary structure. If the overflow structure fills up, only those queues in the overflow structure are affected.

## System Logger

As already mentioned, CQS does not have its own internal logger. It uses the services of the System Logger (IXGLOGR) to log any changes to the shared queue structures. The System Logger is a component of z/OS and executes in a single address space in each z/OS image. All users of System Logger services (for example, CQS) in a system communicate with the same Logger address space. When CQS wants to write a log record for a shared queue structure change, it issues an IXGWRITE request to the System Logger, which then externalizes the log record to the *log stream* identified on the write request. In an IMS shared queues environment, this log stream always exists in a *logger structure* in the Coupling Facility. In order to allow the sharing of the log streams across multiple z/OS images, CQS does not support the use of DASD-only log streams.

Details of the System Logger can be found in Chapter 2, "System Logger fundamentals" on page 7, and in *z/OS MVS Setting Up a Sysplex*, SA22-7625.

### **Log streams**

A log stream is a sequential series of log records written by the System Logger at the request of a log writer such as CQS. They are written in the order of their arrival and may be retrieved sequentially, forward or backward, or uniquely, by log token (key). For an IMS shared queues group, there may be two log streams - one for full function and another for fast path EMH. This is true even if the shared queues are split across a primary and an overflow structure. The more recent log stream records will reside in *interim storage* in a Coupling Facility list structure (the logger structure). Older records may be offloaded to *permanent storage* on DASD (offload data sets). Eventually CQS will delete log records from the log stream when they are no longer needed for message queue recovery. When this occurs, the logger structure space, or the DASD space, used to hold these log records is freed up.

### **System Logger structures**

IMS shared queues always uses Coupling Facility list structures for its log streams. A DASD-only log stream is not an option for the interim storage of the log stream since the staging data sets cannot be shared across the sysplex by other members of the shared queues group. There is one logger structure for the full function log stream and another for the fast path EMH log stream (if it exists). The logger structure is defined in the CFRM policy and is allocated as a *persistent* list structure. Persistence means that the logger structure is retained even when there are no connected users. Although log streams can share a logger structure, IMS does not recommend placing the full function and fast path EMH log streams in the same structure. Each log stream would be allocated one half of the available space in the structure and it is not likely that the characteristics of the full function and EMH workloads would be similar.

### **Offload data sets**

When the logger structure reaches a user-specified (or default) *high offload threshold*, one of the System Loggers in the sysplex will offload all or part of the log stream from the logger structure to *permanent storage* on an offload data set. The offload data sets are VSAM linear data sets. By default, up to 168 offload data sets may be allocated per log stream. Once in an offload data set, the log records remain there until deleted by the CQS. Since the log stream can potentially become quite long, you may define, in the LOGR policy, multiple extents to contain the offloaded log stream. Each extent consists of another 168 data sets. It is very unlikely that CQS would require more than a single extent of 168 offload data sets.

### **Staging data set or data space**

The System Logger on each z/OS image will always *duplex the interim storage* portion (that part in the logger structure) of a log stream. One copy of interim storage goes in the logger structure. You have a choice about where the second copy goes. It may go to a data space, or you may choose to keep the second copy in a data set called the *staging data set*. All log records which exist in the logger structure also exist in one of these two places. As soon as they are deleted from the structure, or offloaded from the structure to an offload data set, they are also deleted from the staging data set or data space. Note that log write requests are synchronous and do not complete until the log record is written to both copies of the interim storage. The use of staging data sets for the duplexed copy of interim storage may be desired or required to provide the necessary level of recoverability such as with ICFs or in a configuration where you are mirroring your DASD to a remote location, but the use of staging data sets may limit the throughput of a given IMS image and should be considered carefully. Some performance information is shown in 4.5, "CQS and System Logger performance considerations" on page 134.

Each System Logger instance has its own staging data set or data space in which it keeps a copy only of what that System Logger put in the structure. However, if there is only one CQS active in the shared queues group, or only one is generating a significant amount of log data, the log data will have been written by one System Logger, so the staging data set size

(defined in the LOGR policy) should be large enough to hold all the log records in the structure. Because of the way space is used in the staging data set, the data set should be much larger than the structure.

### Log stream size and relationships

Figure 4-2 shows that portions of the log stream may exist in both the logger structure and on the offload data sets. The beginning of the log stream (the oldest active record) is on the offload data set containing the log records created since the older of the two structure checkpoints was taken. The end of the log stream is in the logger structure. When CQS takes a new structure checkpoint, it will delete log records from the beginning of the log stream, establishing a new beginning just after the older structure checkpoint. The first log record at the new beginning will be a “structure checkpoint begin” (type x’4201’). This will be for the older of the two structure checkpoints.

When sizing the logger structure (defined in the CFRM policy) and the offload data sets (defined in the LOGR policy), you must allow for enough space in the log stream to contain all the log record entries since the *older structure checkpoint*. The size of the structure determines how frequently the System Logger will offload, but the size of the offload data sets determines how much data can exist in the log stream.

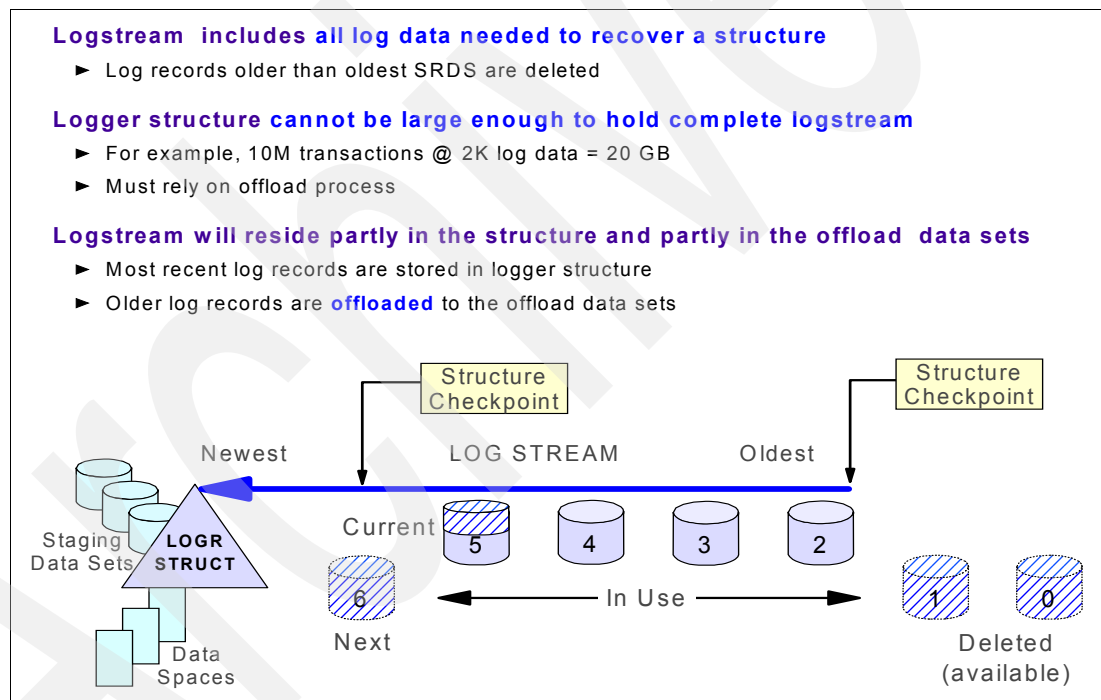


Figure 4-2 Where is the log stream?

The log stream shown in Figure 4-2 currently exists in the structure and four offload data sets. The next offloads will complete filling offload data set “5” and then allocate number “6”. Data sets “0” and “1” contain data that has been logically deleted, however the data sets will not be physically deleted until the System Logger goes through allocation processing for a new offload data set. There can be only 168 offload data sets allocated at one time, although the sequence numbers in the data set names do not reset to “0”.

## 4.1.2 How CQS uses the System Logger

IMS may receive messages from several sources, including the network or an application program inserting to the message queues. Before processing that message, IMS must first



queue it in the shared queue structure where it is available to all IMSs in the shared queues group.

## Message logging

Figure 4-3 shows how message and log data flow during the receiving and enqueueing of an IMS input transaction. Similar functions occur for output messages. Every activity that changes the shared queue structure is logged by CQS.

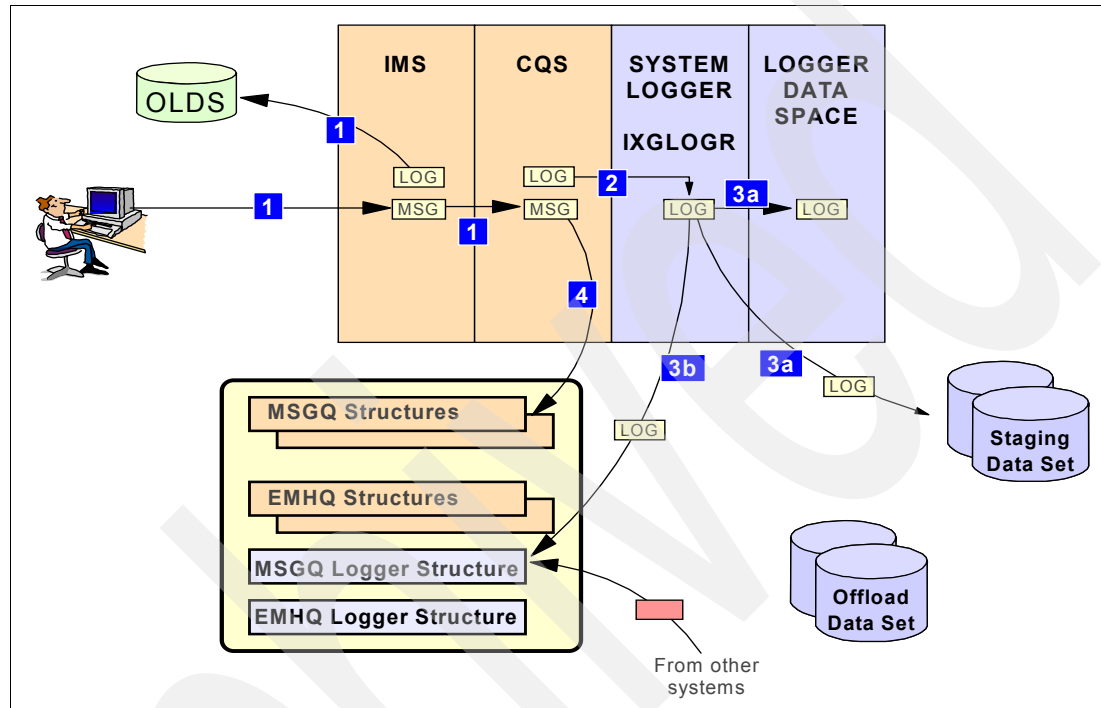


Figure 4-3 Flow of message and log traffic

The flow is:

1. When these messages are received in the local IMS *message queue pool* (QPOOL), they are logged to the IMS OLDS, after which a request is made to CQS to PUT the message on the shared queue. That IMS task (ITASK) then waits for CQS to accept the message and respond.
2. When CQS gets the PUT request from IMS to put a message on the shared queue, CQS creates a log record in a local CQS buffer and issues an IXCWRITE request to the System Logger to write the log record to the appropriate log stream. CQS then waits for acknowledgement that it has been logged before continuing with the original PUT request from IMS.
3. When the System Logger gets the IXCWRITE request from CQS, it writes the log record to the data space or staging data set (3a), then writes it to the LOGR structure (3b). When it successfully completes this process, it acknowledges positively to CQS.
4. When the System Logger acknowledges to CQS that it has successfully logged the data, CQS PUTs the message on the shared queue structure and responds to IMS. When CQS responds that it has successfully queued the message, IMS deletes that message from its local QPOOL.

## Shared log stream

There are multiple IMSs and multiple CQs in a shared queues group. But there is only one shared queue structure, and therefore only one log stream with updates to that shared queue structure. Every CQS in the IMSplex must use the same log stream. Each CQS writes update log records to that log stream and the System Loggers interleave those log records on the log stream in the order in which they are received—that is, in time sequence.

### Log writes

Figure 4-4 shows how log records are interleaved on the log stream by the System Logger in the order in which they are written by the CQS clients.

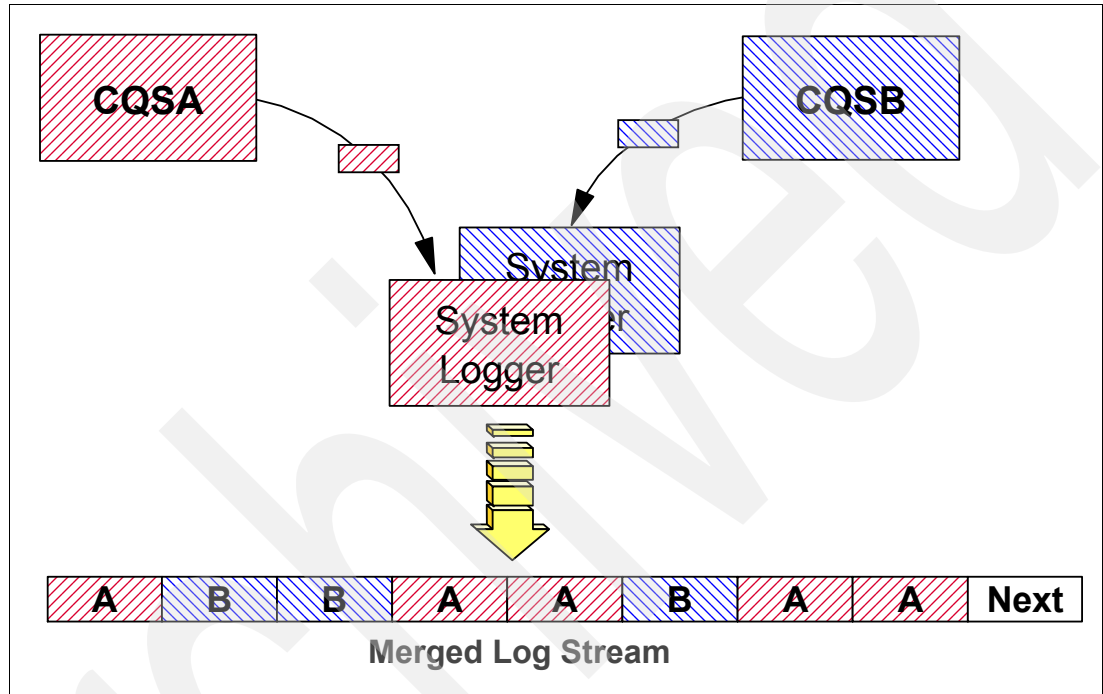


Figure 4-4 Shared log stream

## Log reads

The log records are needed by CQS when it detects that a shared queue structure has failed. When this happens, the first CQS to detect the failure is responsible for recovering the structure using the SRDS and all the log records from the time of the *most recent complete structure checkpoint*. Figure 4-5 shows that one CQS can recover the message queue structure by reading the merged log stream. All log records from all CQSs are returned to CQSA.

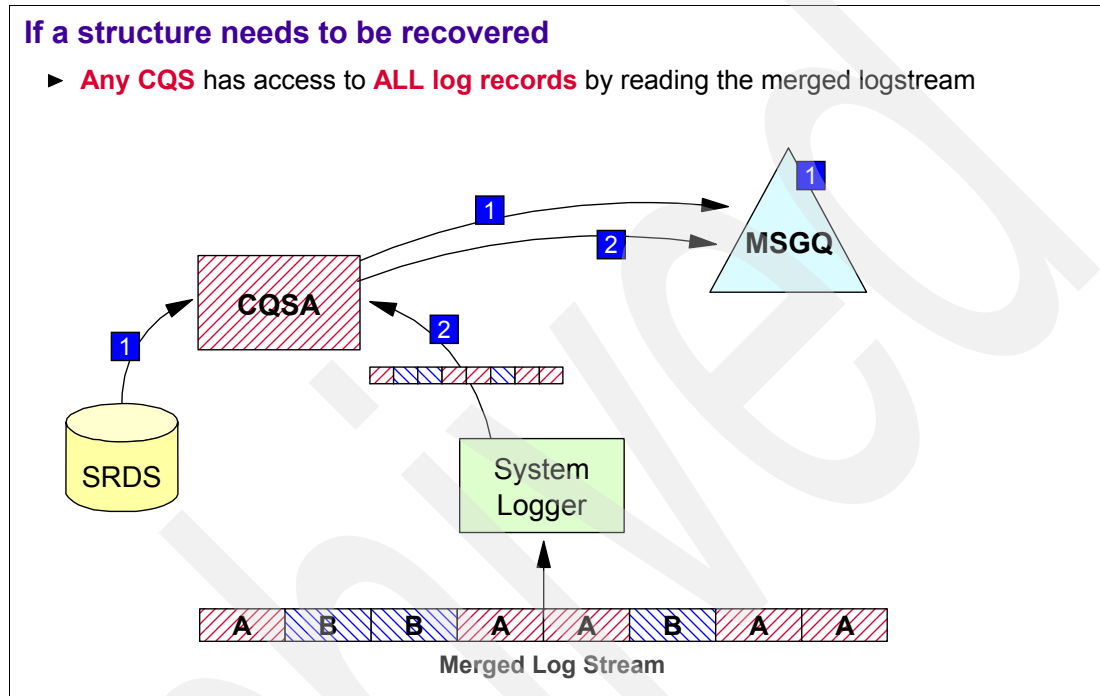


Figure 4-5 Reading the log stream

The process is:

1. When CQS discovers that a shared queue structure needs to be recovered, it allocates a new structure in the Coupling Facility, reads the SRDS, and restores the structure to its condition at the time the structure checkpoint was taken.
2. CQS then issues IXGBRWSE requests to the System Logger to read the log stream, beginning with the first log record created after the structure checkpoint was taken, and continuing to the most recent log record. As each log record is read, it is used to update the shared queue structure. Although part of the log stream may be on DASD, a CQS reading the log stream merely has to issue IXGBRWSE requests to the System Logger to retrieve all required log records.

## Log records

In a shared queues environment, both IMS and CQS create log records with similar functions. For example, IMS logs an input message in a type x'01' log record, and an output message in a type x'03' log record. When IMS requests CQS to put either type message on the shared queue, CQS logs it as a type x'07' log record. When IMS deletes a message from its local queues, it writes a type x'36' log record. CQS writes a x'0D' log record when it deletes a message from the shared queue. Both IMS and CQS logging occur in a shared queues environment.

### Log record types

There are 13 CQS log record types, each with one or more subtypes, that CQS uses to manage itself and the message queue structures. For a description of the CQS log record types, see *IMS Common Queue Server Guide and Reference*, SC27-1292. The CQS log record DSECTs can be obtained by assembling macro CQSLGREC TYPE=ALL from SDFSMAC (IMS macro library). For the IMS log record DSECTs, assemble macro ILOGREC RECID=ALL in the same library. A description of these log records can be found in *IMS V8 Diagnosis Guide and Reference*, LY37-3742.

### Printing the log stream

The IMS program *File Select and Formatting Print Utility*, DFSERA10, can be used to print a CQS log stream. Example 4-1 shows a sample job that will print all the log records in the log stream. The CQSERA30 exit formats the log records for you. Since the log stream is probably protected by RACF, the user ID will have to be given access. See 4.2.3, “Security definitions” on page 130 for an example of how this is done. The utility is documented in *IMS V8 Utilities Reference: System*, SC27-1309.

#### Example 4-1 Sample job for printing the log stream

---

```
//CQSERA10 JOB ....
//STEP1 EXEC PGM=DFSERA10
//STEPLIB DD DSN=IMSPLEX0.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=A
//TRPUNCH DD SYSOUT=A,DCB=BLKSIZE=80
//SYSUT1 DD DSN=MSGQ.LOG.STREAM,
// SUBSYS=(LOGR,IXGSEXIT,'FROM=(2007/087,07:00:00),TO=(2007/087,23:30:00),LOCAL'),
// DCB=BLKSIZE=32760
//SYSIN DD *
CONTROL CNTL H=EOF
OPTION PRINT EXITR=CQSERA30
END
/*
```

---

### 4.1.3 System Logger offload processing

Eventually the logger structure would become full, since IMS never stops putting messages on the shared queue and CQS never stops writing log records. There is a way to shorten the log stream by deleting log records no longer needed, which we will discuss later. But since this is not usually done frequently enough to keep the log stream small enough to fit in a logger structure, the System Logger must periodically copy the older log records to a DASD data set and delete them from the logger structure. This process is called *offload processing* and is driven based on a user-defined high offload threshold percentage.

Figure 4-6 shows, at a high level, the offload process. Note that this is strictly a System Logger process and that CQS is not involved in any way. CQS and System Logger activity is not quiesced during the offload process.

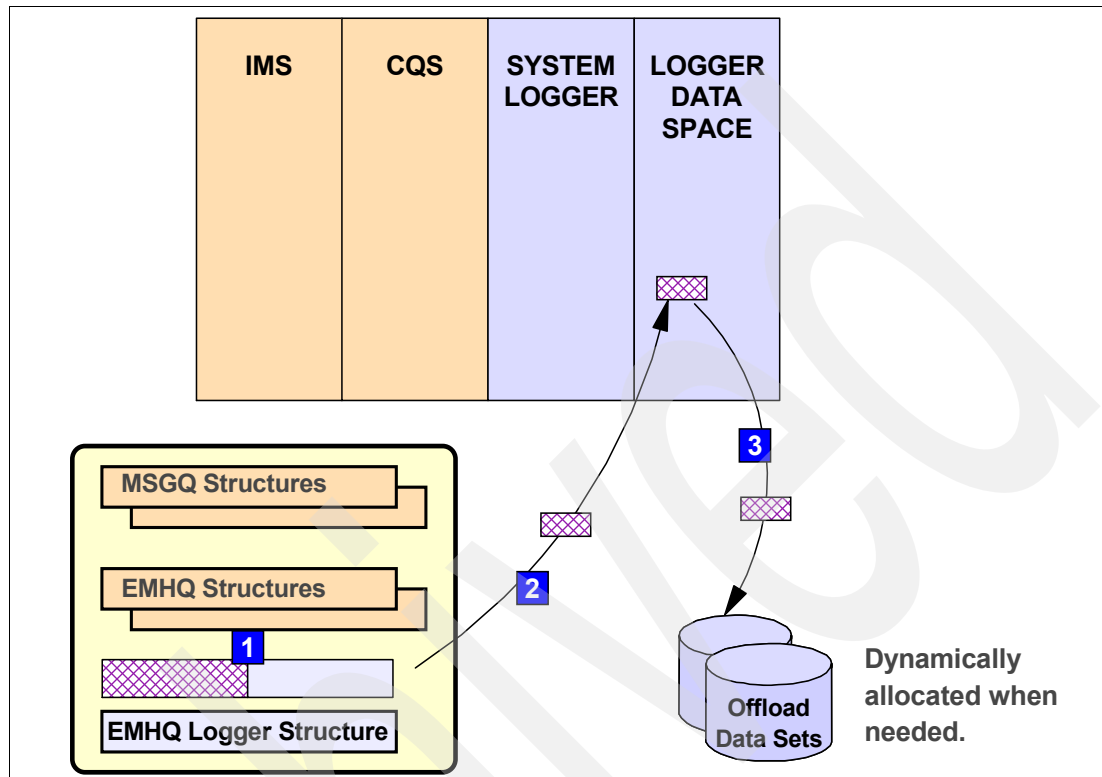


Figure 4-6 Offloading the log stream

When the System Logger detects that the logger structure has reached the *highoffload* threshold, it reads enough log records from the log stream to reduce the logger structure utilization to a user-defined *lowoffload* threshold. Before deleting them (freeing up the space in the structure), they are written to one (or more) DASD logger *offload data sets*. They will also be deleted from the duplex copy (data space or staging data set).

#### 4.1.4 Offload data set deletion

An offload data set is *eligible for deletion* when it contains no active log record entries, that is, when CQS has deleted all log records in that data set. However, physical deletion does not occur until the *next offload data set allocation*. If this never occurs, then those data sets eligible for deletion may never get deleted, occupying DASD space without performing any useful function. A more likely scenario is that CQS deletes enough log records to bring the log stream entirely within the logger structure. This makes all offload data sets eligible for deletion. If, however, due to low activity or frequent structure checkpoints, allocation of another offload data set is never done, or not done for a long period of time, those data sets will not be physically deleted, or not for a long time. It is therefore of some importance to set the size of these data sets (LS\_SIZE parameter in the LOGR policy) large enough so that you never run out of available extents (168), but not so large that a lot of DASD space is consumed with deleted log data.

## 4.1.5 Importance of the log stream data

Unlike other subsystems like CICS, CQS only requires the log stream to recover the shared queue structure if it fails (structure failure, Coupling Facility failure, or loss of connectivity from all systems). It may be used by CQS to restart, but it is not required. If the logger structure fails, or becomes full, or an IXGWRITE request to the System Logger fails, CQS immediately initiates a structure checkpoint, eliminating the need for any log records. Of course CQS cannot perform any more shared queue structure services until the log stream is available, but there is no loss of message queue data unless there is a double failure - that is, if both the log stream *and* the shared queue structure fail at the same time. Even though the System Logger has its own means of log stream recovery, it is usually wise to keep the two structures on different Coupling Facilities to avoid a single point of failure.

## 4.2 Defining the System Logger in the IMSplex

There are several definitions required in CQS and in z/OS to implement CQS use of the System Logger. There are no System Logger-related definitions in IMS itself.

### 4.2.1 Subsystem (CQS) definitions

CQS definitions of the log streams are contained in CQS PROCLIB member CQSSGxxx (the CQS *global* PROCLIB member). Only the *log stream name* is defined - not the logger structure name, nor any other characteristics of the log stream. Example 4-2 is an example of a CQSSGxxx PROCLIB member for a CQS with both a full function shared queue and a fast path EMH shared queue.

*Example 4-2 Sample log stream definition in CQS*

---

```
STRUCTURE(STRNAME=MSGQ.STRUCTURE
OVFLWSTR=MSGQ.OVFLW.STRUCTURE
LOGNAME=MSGQ.LOG.STREAM
SRSDSN1=CQS.MSGQ.SRDS1
SRSDSN2=CQS.MSGQ.SRDS2)
STRUCTURE(STRNAME=EMHQ.STRUCTURE
OVFLWSTR=EMHQ.OVFLW.STRUCTURE
LOGNAME=EMHQ.LOG.STREAM
SRSDSN1=CQS.EMHQ.SRDS1
SRSDSN2=CQS.EMHQ.SRDS2)
```

---

Note that there is only one CQSSGxxx global PROCLIB member in the shared queues group. All CQSs use the same member (or members which are identical). The LOGNAME parameter must match the LOGSTREAM NAME parameter in the LOGR policy.

### 4.2.2 System Logger definitions

System definitions for the log stream and logger structure are in two places. The log stream and its characteristics are defined in the LOGR Policy in the LOGR Couple Data Set (CDS). LOGR list structure definitions are in both the CFRM CDS (defines the structure to XES), and the LOGR policy (defines the relationship between the log stream and the structure).

#### LOGR structure definition

When a Coupling Facility list structure is used as interim storage for a log stream (always the case for CQS), then that structure must be defined in the active CFRM policy. The policy defines the name and size of the structure, as well as some other characteristics.

Example 4-3 is an example of the definition of the LOGR structure for the full function message queues in the CFRM policy. A similar definition would be required for the EMH queue logger structure.

*Example 4-3 Sample LOGR structure definition*

---

```
DATA TYPE(CFRM)
  DEFINE POLICY NAME(POLICY1)
    STRUCTURE NAME(MSGQ.LOGR.STRUCTURE)
    SIZE(100000)
    INITSIZE(50000)
    DUPLEX(ALLOWED)
    ALLOWAUTOALT(NO)
    REBUILDPERCENT(1)
    PREFLIST(CF01,CF02)
    EXCLLIST (MSGQ.STRUCTURE)
```

---

The meanings of these parameters are briefly described below with considerations for CQS and IMS shared queues. For a more complete description of the parameters themselves, see Chapter 2, “System Logger fundamentals” on page 7, and *z/OS MVS Setting Up a Sysplex*, SA22-7625.

► **NAME**

Defines the name of the CFRM policy. This policy must be active for the definitions to be effective. The CFRM policy is activated using the command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=POLICY1
```

► **INITSIZE**

This is the initial size of the structure, in 1 KB increments. Since this structure will almost always become full before a structure checkpoint is taken, the actual size does not matter too much. Eventually, all log records in the structure will have to be offloaded regardless of the size. A smaller structure means that less memory is needed in the Coupling Facility and in the central processor (CPC) to duplex the structure in a data space. If you have memory constraints, making it smaller will require more frequent offloads, but have little effect on the overall performance. Making it larger will reduce the number and frequency of offloads, but in the end, the same amount of data will be offloaded. You might also use a target offload frequency (for example, once every 30–60 seconds during peak transaction volumes) to set the size and HIGHOFFLOAD values (described below). A 50 MB logger structure size is suggested as a reasonable size at which to begin. If the structure needs to be made larger to reduce the number of offload processes, the structure size can be altered up to its maximum size set using a SETXCF START,ALTER command.

► **SIZE**

This is the maximum size to which the structure can be altered without updating the CFRM policy and rebuilding the structure. Note that, unlike CQS and the message queue structures, the System Logger *does not alter the size* of the structure from its current size. It must be done by the user using the command:

```
SETXCF START,ALTER,STRNM=structure-name,SIZE=new-size
```

► **DUPLEX**

Indicates whether the structure may be duplexed, using System-Managed Duplexing, for availability. ENABLED means that the structure is automatically duplexed when it is allocated. ALLOWED means that it is not done automatically by XES, but can be turned on by the operator if desired. Duplexing of the logger structure can have significant

performance consequences. See the performance section later in this chapter for more information. The following command may be used to enable duplexing:

```
SETXCF START,REBUILD,DUPLEX,STRNAME=MSGQ.LOGR.STRUCTURE
```

▶ **ALLOWAUTOALT**

Indicates whether the system may automatically alter the structure size or entry-to-element ratio. This should *always* be set to NO for logger structures, since the System Logger monitors and alters the structure attributes itself. Allowing the system to automatically alter the structure size may produce undesired results.

▶ **REBUILDPERCENT**

Indicates that if connectivity to the structure is lost by more than this percentage, it should be rebuilt on another Coupling Facility where the connectivity is better. This should be set to 1 (percent) or allowed to default so that it will always be rebuilt on another Coupling Facility if connectivity from one of the systems fails.

▶ **PREFLIST**

Identifies the Coupling Facilities which are candidates to contain this structure. For rebuild or duplexing to work, there must be at least two.

▶ **EXCLLIST**

Requests that the structure not reside on the same Coupling Facility as the structure(s) named in the parameter. We recommend that you put the logger structure on a different Coupling Facility than the shared queue structure to avoid a single point of failure. This does not prevent the logger structure from being allocated on the same Coupling Facility if there is no other CF available.

Note that there is nothing in this definition to indicate that the structure is a list structure, or that it is to be used by the System Logger (unless you interpret this from its name). The name (MSGQ.LOGR.STRUCTURE) must match the name defined in the LOGR policy.

## LOGR policy definition

The log stream, and the structure used by the log stream, is defined in the LOGR policy. The LOGR policy relates the log stream with the logger structure, and defines the characteristics of the log stream and structure. Example 4-4 shows an example of the portion of the LOGR policy defining the CQS log stream and logger structure for the full function message queue (MSGQ) structure.

*Example 4-4 Sample LOGR policy definition*

---

```
DATA TYPE(LOGR)
  DEFINE STRUCTURE
    STRUCTNAME(MSGQ.LOGR.STRUCTURE)
    AVGBUFSIZE(512)
    MAXBUFSIZE(65276)
    LOGSNUM(1)

  DEFINE LOGSTREAM
    NAME(MSGQ.LOG.STREAM)
    STRUCTNAME(MSGQ.LOGR.STRUCTURE)
    HLQ(IMSPLX0)
    STG_DUPLEX(YES)
    STG_SIZE(50000)
    STG_DATACLAS(STAGING)
    DUPLEXMODE(COND)
    LOGGERDUPLEX(COND)
    LS_SIZE(256000)
    LS_DATACLAS(OFFLOAD)
```



HIGHOFFLOAD(50)  
LOWOFFLOAD(0)  
AUTODELETE(NO)  
RETPD(0)  
Other SMS parameters

---

It is important that the LOGR structure be defined in the (active) CFRM policy before the LOGR policy is updated to contain your new log streams. It is also necessary to define, in the LOGR policy, the logger structure first and then the log stream. The following are short descriptions of the parameters in the LOGR policy. For a more complete description, see Chapter 2, "System Logger fundamentals" on page 7 and *z/OS MVS Setting Up a Sysplex*, SA22-7625.

### DEFINE STRUCTURE

You must define the logger structure to be used by the log stream. The following parameters are used by the System Logger to set some of the characteristics of the logger structure when it is first allocated (first connect time). Not all possible parameters are shown in this example.

► **STRUCTNAME**

This name must match the name in the CFRM policy and in the log stream definition to follow.

► **AVGBUFSIZE**

This parameter is the average size of a (CQS) log record and is used by the System Logger to set the initial entry-to-element ratio. Only a few CQS log records log the content of the message. Many only log the fact that a message has been read and moved to another queue, or has been deleted. Some log system events, like structure checkpoint started or ended. Most of these other log records tend to be small but numerous and can make the average size of a log record small. This is a difficult number to predict but the good news is that the System Logger will monitor this ratio every 30 minutes and adjust the ratio if necessary. If you are already running shared queues in production or test (with transactions similar to production) you may use the logger utility (IXCMIAPU) report function or the IXGRPT1 program to find the average buffer size. If this information is not available, we recommend that you set this value to 512 (bytes) and let the System Logger adjust it if necessary.

► **MAXBUFSIZE**

This value determines the maximum size log record that can be written to the log stream. It also determines the size of the data elements in the structure. Any value less than or equal to 65276 will cause the System Logger to allocate the structure with 256-byte data elements. Values over this will result in 512-byte data elements and should be avoided (waste of space). For IMS full function or fast path EMH, there is no reason to specify any value other than 65276.

► **LOGSNUM**

This determines how many log streams can share the LOGR structure. For CQS, this number should always be one. You should not share a LOGR structure with both the full function and fast path EMH log streams. Their usage of structure resources is likely to be very different.

## DEFINE LOGSTREAM

The following parameters define the characteristics of the log stream. Not all possible parameters are shown in this example.

► NAME

This is the name of the log stream. It is the only name known to CQS and must be the same as the LOGNAME parameter in CQSSGxxx.

► STRUCTURENAME

This is the name of the logger structure that this log stream will reside in. It must be the same as the name defined in the DEFINE STRUCTURE command in this LOGR policy, and must have been defined in the CFRM policy.

► HLQ

Defines a high level qualifier for the log stream offload data set names and the staging data set names. In the above example, the staging data set name and the offload data set names will have a high level qualifier of IMSPLEX0.

► STG\_DUPLEX

This determines whether the interim log stream will be duplexed to a staging data set or to a data space. Duplexing to a data space is generally the most desirable from a performance perspective. However, staging data sets may be necessary for highest availability, especially with ICFs or with DASD mirroring. With Internal Coupling Facilities you probably want to specify STG\_DUPLEX(YES) and DUPLEXMODE(COND). For a DASD mirroring environment, you would specify STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND).

► STG\_SIZE

The System Logger will always duplex the log stream, but you have the choice of whether to duplex it in a data space or on DASD in a *staging data set*. This parameter sets the size of the staging data set in 4 KB increments. Because of the way space is used in the staging data set (see 8.2.2, "Sizing interim storage" on page 275), this should be several times the size of the structure to avoid unnecessary offloads, and to allow maximum use of the allocated structure size. Note that the structure size specified in the CFRM policy is in 1KB increments.

► STG\_DATACLAS

This parameter specifies the name of the SMS data class that is to be used when allocating the staging data set. Staging data sets should have VSAM share options set to (3 3), so the data class specified on this parameter should have those share options specified. Additionally, the CI Size must be 4 KB.

► DUPLEXMODE

This parameter is used when STG\_DUPLEX(YES) is set and determines the conditions under which the log stream should be duplexed to the staging data sets.

DUPLEXMODE(COND), for conditional, means to use the staging data sets only if the system's connection to the Coupling Facility contains a single point of failure. A single point of failure exists if the CF is volatile (no power backup) or if it resides on the same CPC as the z/OS system connecting to it. We recommend setting this value to COND, which will use the staging data sets *only* if the data space and the structure are failure dependent unless you are using DASD mirroring, in which case you would want UNCOND.

► LOGGERDUPLEX

This parameter determines whether the System Logger will use its own duplexing method (staging data set or data space) if the structure is being duplexed in the Coupling Facilities. COND means that if the *structure* is duplexed, and the two structure instances

are failure isolated from each other, then the System Logger will not duplex to either the staging data set or a data space. Since the default is UNCOND, you may want to code this parameter as COND if the logger structure is being duplexed.

► LS\_SIZE

This parameter sets the size of the offload data set in 4 KB increments. At a minimum, the size should be set to a value large enough that you don't run out of your default maximum allocations (168) across two structure checkpoints. This is also a difficult number to optimize, but can be estimated by calculating the amount of data logged between structure checkpoints. This is dependent on the transaction rate and the amount of data logged per transaction. As an example, if you are running 1000 transactions per second, and log 2000 bytes per transaction (input plus output message length and a little bit more), then you are logging (about) two megabytes per second. This equates to 7.2 GB per hour. Since you need enough space for two structure checkpoint intervals, for every hour between checkpoints you need at least 14.4 GB of total offload data set space. This is in the neighborhood of 90 MB per data set, or LS\_SIZE=22500. This should be considered the *absolute minimum* to contain the entire log stream. This is not a number that you want to fine-tune. Make the offload data sets much larger than the minimum to avoid the possibility of the log stream filling up. It uses no more DASD space. It just uses fewer, but larger, data sets.

► LS\_DATACLAS

This parameter specifies the name of the SMS data class that is to be used when allocating the offload data sets. Offload data sets should have VSAM share options set to (3 3), so the data class specified on this parameter should have those share options specified. This data class should be set up with a CI Size of 24 KB, rather than the 4 KB that must be used for the staging data sets.

► HIGHOFFLOAD

This parameter determines when the offload process is driven. When the percentage of data elements in use in the structure reaches this value, the offload process is driven. This number is not critical except that it should probably be between 50% and 80%. Do not go above 80%, as this increases the risk of the structure filling before the offload can free up space in the structure (see 8.4.1, "High and low offload thresholds for funnel-type log streams" on page 284). You should be sure to leave enough room for logging to continue without running out of space while offload is processing.

► LOWOFFLOAD

This parameter determines how much is offloaded. For CQS, this should be set to zero (0%). That means that offload should move all data entries on the structure at the time it is driven. New data entries that arrive after offload begins are not offloaded until the next cycle. Since CQS uses this data only for forward recovery of a message queue structure, which should be rare or never, there is no reason to leave any data in the structure.

► AUTODELETE

*Never* specify YES for this (NO is the default). It may cause the System Logger to automatically delete offloaded data before CQS deletes it.

► RETPD

This could be used if you have some reason for keeping CQS log records beyond the time necessary for recovery. If so, you can specify this as the minimum number of days to keep a log record entry before deleting it, even though CQS may have already issued a delete for it. Setting the retention period to zero (0) or letting it default, means that deletion will be controlled strictly by CQS.

There are other parameters that can be specified in the LOGR policy, but most are not of interest to CQS or are determined by installation standards. Even some of the above may be determined by installation standards.

### 4.2.3 Security definitions

The log stream will be a protected resource defined in the *Security Access Facility (SAF)* LOGSTRM class. Each CQS user ID must be permitted UPDATE access to the log stream. If you have any users who will be just reading the log stream, such as DFSERA10, they need READ access.

*Example 4-5 Sample log stream security definition*

---

```
RDEFINE LOGSTRM MSGQ.LOG.STREAM UACC(NONE)
PERMIT MSGQ.LOG.STREAM CLASS(LOGSTRM) ID(cqsuserid1) ACCESS(UPDATE)
PERMIT MSGQ.LOG.STREAM CLASS(LOGSTRM) ID(cqsuserid2) ACCESS(UPDATE)
PERMIT MSGQ.LOG.STREAM CLASS(LOGSTRM) ID(cqsuserid3) ACCESS(UPDATE)
PERMIT MSGQ.LOG.STREAM CLASS(LOGSTRM) ID(readuser1) ACCESS(READ)
PERMIT MSGQ.LOG.STREAM CLASS(LOGSTRM) ID(readuser2) ACCESS(READ)
```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

## 4.3 CQS and System Logger operational considerations

There are not very many operational considerations for CQS's use of the System Logger. If there are no errors, and if the log stream doesn't get full, then it pretty well takes care of itself. Even some errors are handled automatically by CQS with little or no (CQS) operations involvement. Most of the operational requirements are the responsibility of systems management.

### 4.3.1 Controlling the use of System Logger

CQS requires that the System Logger be available, and that a log stream be defined for full function shared queues and one for fast path EMH shared queues (if fast path is defined in your environment and you are pre-V9). If the log stream name is not defined in CQSSGxxx, CQS will fail during initialization. If the System Logger is not available, CQS will issue a message (CQS0350W) and wait.

### 4.3.2 Defining log streams dynamically

CQS does not dynamically define log streams and it does not use model log streams. The CQS log streams must be defined in the LOGR policy before CQS is started.

### 4.3.3 Displaying log stream information

There is an MVS command that displays information about the status of the System Logger or about the log stream connections. The command is:

```
D  LOGGER,STATUS
D  LOGGER,CONN,xxxx
```

Where 'xxxx' is zero or more parameters defining the output of the command. This command does not display the current utilization statistics of the log stream. It just tells you the status of

the log streams and what structures are defined for them. 4.3.4, “Monitoring for offload conditions” on page 131, describes a better way to get useful information.

“Printing the log stream” on page 122 shows a sample job that can be used to print the log stream itself. All normal DFSERA10 control statements apply.

### 4.3.4 Monitoring for offload conditions

Unless additional extents are requested by the DSEXTENTS parameter in the LOGR policy, the log stream will become full when there are 168 offload data sets allocated and all are full, and interim storage is full. If this occurs, you will get a LOG FULL condition. CQS issues a message, deletes a few log records from the log stream to make space for structure checkpoint log records, takes a structure checkpoint, and continues processing. The CQS message is:

```
CQS0350W CQS LOG CONNECT DS DIRECTORY FULL LOG STREAM logstr-name STRUCTURE str-name
```

While the structure checkpoint relieves the shortage, it will have a temporary impact on the availability of the log stream and CQS will not be able to process any IMS PUT requests until the structure checkpoint is complete and CQS has deleted the tail of the log stream. To avoid this problem, you can run the IXCMIAPU utility periodically to determine how many offload data sets are in use. If you find that the number of offload data sets is close to 168, you should either increase the size of the offload data sets (LS\_SIZE parameter) so that you don't need so many offload data sets, or format a new LOGR CDS with a larger DSEXTENT value, which will allow you to have more than 168 offload data sets.

Example 4-6 shows a sample job that will let you monitor the log stream offload data set usage. The output of this job reports information about the log streams and the logger structure itself. Among other information, it tells you exactly how many offload data sets are currently in use. It also provides structure information, such as what average buffer size was specified and what the effective buffer size really is. In this example, the System Logger will have changed the entry-to-element ratio.

*Example 4-6 Sample job to list log stream information*

---

```
//MIAPU JOB ....
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(NO)
  LIST LOGSTREAM NAME(MSGQ.LOG.STREAM) DETAIL(YES)
  LIST STRUCTURE NAME(MSGQ.LOGR.STRUCTURE) DETAIL(YES)
/*
```

\*\*\*\*\* SAMPLE OUTPUT \*\*\*\*\*

```
LOGSTREAM NAME(MSGQ.LOG.STREAM) STRUCTNAME(MSGQ.LOGR.STRUCTURE) .....
.....<logr policy parameters for this log stream>
....
.... <other detailed information about the log stream and offload data sets>
....
```

**NUMBER OF DATA SETS IN LOG STREAM: 43**

```
STRUCTURE NAME(MSGQ.LOGR.STRUCTURE) LOGSNUM(1)
MAXBUFSIZE(65272) AVGBUFSIZE(512)
EFFECTIVE AVERAGE BUFFER SIZE(1024)
```

---

In the beginning, you should do this often enough to get an idea of how long it takes to fill an offload data set during peak processing periods. You can use this information to adjust the size of your offload data sets (LS\_SIZE) and your structure checkpoint frequency.

## 4.4 CQS and System Logger error recovery considerations

CQS is dependent on the availability of the log stream in order to provide shared queue services to its IMS client(s). When CQS connects to the log stream, or when CQS issues a read or write request, the System Logger may return a non-zero return code and reason code, indicating some type of problem.

### 4.4.1 Non-zero return and reason codes from the System Logger

When the System Logger returns a non-zero return code to a CQS request, the reason code indicates the problem. Messages issued by CQS are described in 4.4.2, “Important messages” on page 133. CQS can take one of the following actions, based on the reason code and whether CQS thinks this is a problem that will soon be resolved:

- ▶ Wait for the System Logger to correct the problem and issue an ENF48 signal. When CQS gets the ENF48, it continues processing. No messages are issued by CQS. This occurs under the following conditions. The reason code returned to CQS is in parentheses after the reason text below, in case you want to explore this further.
  - Log stream structure full. Offload in progress (860)
  - Logger CF rebuild in progress (861)
  - Logger structure rebuild in progress (862)
  - Logger structure or CF failed (863)
  - No connectivity to CF (864)
  - Staging data set full (865)
  - System Logger address space initializing (891)
- ▶ If the System Logger is not available (890) when CQS tries to connect, or becomes unavailable after CQS has already connected, CQS will issue a message (CQS0350W) and wait for the System Logger to start and send an ENF48 signal.
- ▶ If CQS receives a bad return and reason code from an IXGWRITE request, it will issue a message indicating the reason, take a structure checkpoint, and continue processing. The possible causes are:
  - Loss of data in the log stream (405)
  - Possible loss of data in the log stream (407)
- ▶ If CQS receives a bad return and reason code from an IXGCONN (connect) request, it will issue a message (CQS0350W) indicating the reason, and then take some action depending on the error as follows:
  - Possible loss of data (407)  
CQS abends with a U0014
  - DS directory full (408) or previous offload error (409)  
CQS takes a structure checkpoint and continues processing
- ▶ If CQS gets a bad return code and reason code during IXGBRWSE (read) processing for structure rebuild, CQS aborts the rebuild and issues message CQS0354E. Since this is usually done only for structure recovery, you may have to redefine and reinitialize the structure.

## 4.4.2 Important messages

There are only a few CQS messages that reference the System Logger. These are briefly described below, but are more fully documented in *IMS V8 Messages and Codes Volume 1*, GC27-1301 where the messages are associated with System Logger return and reason codes. The CQS message numbers range from CQS0350W through CQS0354E.

### **CQS0350W *reason* LOG STREAM *logstrname* STRUCTURE *strname***

This message is issued to indicate a variety of error conditions. The *reason* is based on the return and reason code from the System Logger and is provided as **text** in the message.

#### ***System Logger unavailable***

If the System Logger is not available when CQS is initialized, or if it terminates while CQS is running, CQS will issue this message with reason SYSTEM LOGGER NOT AVAILABLE. In this case, CQS cannot do any processing (meaning IMS cannot perform any message queue activity). You should immediately restart the System Logger. CQS will wait for the ENF48 signal from the System Logger, then attempt to connect again.

#### ***CQS log connect possible loss of data***

CQS abends with either a U0014 (if this condition occurs during CQS initialization) or a U0104 (if it occurs any other time). When this happens, delete and redefine the log stream, restart CQS, and take a structure checkpoint.

#### ***CQS log connect previous offload error***

CQS takes a structure checkpoint to establish a recovery point, issues a delete call to the System Logger to delete log records no longer needed for recovery, and continues processing.

#### ***CQS log connect DS directory full - or - CQS log connect previous offload error***

CQS takes a structure checkpoint and issues a delete call to the System Logger to delete log records no longer needed for recovery, and continues processing.

#### ***CQS log write loss of data - or - CQS log write possible loss of data***

CQS takes a structure checkpoint and continues processing. The structure checkpoint establishes a recovery point for the shared queue structure that will not require the (possibly) lost data.

### **CQS0352E LOG WRITE ERROR *reason***

#### ***Due to buffer size***

CQS tried to write a record to the log stream that exceed the MAXBUFSIZE value in the LOGR policy. CQS continues processing. To correct the problem (that is, to be able to write the larger log records), all CQSs must be shut down to disconnect from the log stream. Then delete the log stream and logger structure definitions in the LOGR policy, and change the LOGR policy with a larger MAXBUFSIZE. Then restart CQS. While this is happening, the message queue structures are not recoverable. Take a structure checkpoint immediately.

### **CQS0353E CQS *variable text***

#### ***Various messages indicating CQS has started or stopped reading***

This message merely indicates that CQS is reading the log stream. This may occur during CQS restart to read the system checkpoint record, or it may occur during message queue structure rebuild (recovery). There will be a STARTED and COMPLETED messages and, if a lot of log records are being read, IN PROGRESS messages.

## **CQS0354E LOG READ FAILED, LOG RECORD COUNT *count***

***Count is the number of records read before the error***

This message indicates CQS log stream read processing failed. If it occurs during CQS initialization, then CQS will abend with a U0018. If it occurs during structure rebuild, then rebuild will terminate. This could mean the message queue structure it was trying to rebuild is unusable and may have to be deleted and reinitialized.

## **4.5 CQS and System Logger performance considerations**

System Logger performance is addressed in general in Chapter 8, “System Logger performance and tuning” on page 273. This section addresses the performance considerations unique to IMS shared queues and CQS.

There are several considerations for CQS with respect to the System Logger. One of the key items is the impact of a structure checkpoint, and therefore how often one needs to be taken. A second key item is the use of the logger staging data sets or duplexing of the logger structure, and how they could impact throughput and response times.

### **4.5.1 Structure checkpoints**

A structure checkpoint is required to establish a new recovery point for the message queues, and to signal to the logger that it may delete log records from the log stream that are older than the oldest structure checkpoint. The frequency of structure checkpoints, together with the amount of data being logged between checkpoints, will determine the amount of DASD space necessary for the offloaded log data.

Before a structure checkpoint can be taken, all access to the shared queue structure must be temporarily quiesced, meaning that the CQS address spaces will not process any message traffic to or from IMS. Since only one of the CQSS performs the structure checkpoint, it is necessary for that CQS to notify others to quiesce, and then to know that they have done so. This process is done by updating the CFRM CDS (using the IXLUSYNC macro) and requires several I/Os per CQS<sup>1</sup>. This is what typically takes the most time for the structure checkpoint process, since it must be done to quiesce operations, and then again to resume them after the checkpoint is complete. The actual reading of the message queues into the data space is normally quite fast, but of course will depend on how many messages are actually in the structure when the checkpoint is taken. The writing of the data to the SRDSs is asynchronous, and therefore does not affect the amount of time the CQSS are quiesced for.

There can be a measurable impact to IMS service every time a structure checkpoint is taken, so you do not want to do this too frequently. This impact could be as short as a few milliseconds, or as long as a few seconds, depending on several factors such as the number of CQS address spaces and the speed of the DASD. On the other hand, you do not want to have so many DASD volumes worth of data that a recovery of the shared queues structure would take a prohibitive amount of time. These factors must be carefully weighed in your environment to establish what is reasonable for your installation. z/OS 1.8 provides an enhancement called CFRM Performance Enhancement Stage 2, which should be enabled to provide optimum performance for the structure checkpoint process.

---

<sup>1</sup> The CFRM performance enhancement in z/OS 1.8 significantly reduces the number of requests to the CFRM CDS for this processing, and is expected to result in a sizeable improvement in structure checkpoint time.



## 4.5.2 Structure checkpoint recommendations

To minimize the frequency of structure checkpoints, you should insure that you have sufficiently large offload data sets (LS\_SIZE in the LOGR policy) to prevent too many new data set allocations, and that there is a sufficient amount of DASD space available to hold all the log data from more than two structure checkpoints. The total amount of space will depend on how much data you log between structure checkpoints. This is a factor of transaction volumes and the amount of data logged per transaction. For example, a system doing 500 transactions per second and logging an average of 4,000 bytes per transaction would generate 2 MB of log data per second, as below:

$$500 \text{ transactions per second} * 4000 \text{ bytes per transaction} = 2 \text{ MB per second}$$

At this logging rate, you will generate 7.2 GB of log stream data per hour. Since you will have all the log data for two structure checkpoint intervals, plus more to allow for the asynchronous deletion of the older data sets, you should be able to calculate the required amount of space in the offload data sets. Of course your transaction volume could be less than the example above, or quite likely it could be significantly more with multiple IMS images in a shared queues group. You must decide how disruptive structure checkpoints are in your environment, especially at peak times, and weigh this against the amount of offload data you can afford to keep and how long it would take to recover the shared queues structure in the unlikely event of a failure. Structure checkpoint impact and time to recover in the event of a failure will vary by workload and by the hardware being used, so there is no substitute for testing and measuring this impact in your own environment.

## 4.5.3 Staging data sets

There may be good reasons to use staging data sets for duplexing of the logger interim storage, but their use can impact both throughput and transaction response time. The best solution, from a performance perspective, is to duplex the logger structure in a data space. In this way the only limit is how fast the logger can write to the CF structure. However, when using Internal Coupling Facilities, volatile CFs, or when in a DASD mirroring configuration, you may need to use staging data sets to provide the appropriate level of recoverability. The use of staging data sets will result in reduced transaction throughput compared to using data space duplexing. This is due to the fact that now we are running at the speed of DASD, instead of (actually in addition to) the CF, and that speed will depend on the type of DASD and whether synchronous remote mirroring is being used, and if so, at what distance. Depending on the throughput needs of any single IMS image, this may or may not be of concern.

CQS will typically issue just a fraction over four IXGWrites to the logger per full function transaction. These writes are synchronous to the transaction process similar to the way IMS Checkwrites are processed, except that each write is done as a separate block by the MVS Logger. This means that the total transaction throughput will be limited to whatever I/O rate the DASD volume can sustain. Striping of the staging data sets can provide some relief in that the I/O rate to any single volume will be reduced, which, for some DASD subsystems may also reduce the I/O response time. However, the total I/O rate across multiple volumes, and thus the transaction rate, does not increase in direct proportion to the number of stripes, so use care in setting expectations.

## 4.5.4 Staging data set recommendations

When sizing the staging data sets you need to consider that the logger will always use 4 KB blocks, while the structure will use either 256 or 512 byte elements depending on the MAXBUFSIZE specified in your logger policy. Depending on your actual average buffer size, you could have a sizable amount of unused space in each block on DASD, so you need to

take this into account when sizing the staging data sets. The logger will initiate an offload process based on the HIGHOFFLOAD threshold being reached on either the structure or the staging data set, whichever is reached first. If you do not account for the unused space in the staging data set (the difference between the actual data size and the 4 KB block), you could end up initiating offload very frequently because you might be reaching the threshold sooner than expected. This is not necessarily a problem unless the usable space left in the staging data set is not enough to keep it from becoming full and cause processing to stop.

Staging data sets, when used, are critical to performance, and as such should be placed on the best performing DASD available. Striping of these data sets may be used to reduce the I/O rate to any given volume, and in some cases reduce the response time to the DASD. This may also allow somewhat higher total I/O rates, which would then accommodate additional transaction volume.

If staging data sets become the limiting factor to transaction throughput, it might be necessary to create additional IMS images on separate LPARs (to get additional logger address spaces). More parallel images should allow for considerable growth potential.

### 4.5.5 Measurements

In order to better understand the impact of staging data sets and mirroring, we performed several measurements with an IMS shared queues environment. An application workload was created to drive IMS, with input and output messages of about 2,000 bytes each. Since we were only concerned about the Transaction Manager side of IMS, the application did not perform any database processing — just one GU for the input message and one ISRT for the reply. The applications ran in full function MPP regions with the Pseudo Wait For Input (PWFI) option, but with a processing limit count of 10 in order to get the 08/07 log records for gathering certain transaction statistics. PWFI is a message region option that may be used to avoid unnecessary scheduling overhead.

When looking at the results of our measurements, you should concentrate on the *absolute* difference in transaction response times from one measurement to the next. Because the transactions did not do any database processing, the response times were very low, meaning that the *relative* impact of configuration changes was very high. If you were to implement the same changes in a production environment, you would expect to see your response times change by similar *milliseconds* to what we observed, not by similar *percentages*.

### 4.5.6 Measurement environment

All measurements were done with two IMS images using shared message queues, running in two separate LPARs. One site (Site1) contained both LPARs, one of the Coupling Facilities, and the primary DASD (IBM DS8300). The DASD were synchronously mirrored (PPRC) to the second site (Site2), 20 km away. In addition to the secondary DASD, the second site also contained the second CF. Most of the CF structures were placed in the CF in Site1, with the CF in Site2 being used to test a System-managed duplex configuration.

We first ran with the logger structure and duplexing of the log stream to a data space to create the base case measurements. We then tried using a single staging data set per log stream, followed by creating multiple (2) stripes for the staging data set. We also made a run using system-managed duplexing for the logger structure. As you will see below, some of the runs were made with different numbers of terminals or message regions. This was done to be sure that there were no artificial bottlenecks in our configuration. Several measurements were made in each configuration to verify that the results were repeatable.

Specifically, the hardware and software were configured as follows:

- ▶ There were two z/OS systems in the data sharing and shared queues groups, each running one IMS subsystem (called IM1A and IM2A). z/OS was at Version 1.8, and the host processor was a z9@. A third system, SC47, was used to run TPNS in order to keep that overhead from impacting the systems being measured.
- ▶ There were two Coupling Facilities running CF Level 14 also on z9 hardware. The CFs were connected to the z/OS images using ISC peer mode links.
- ▶ PPRC was used to mirror the DASD that was an IBM 2107-922 (DS8K).
- ▶ IMS Version 9 was used with both full function shared queues and fast path shared EMH enabled, although only full function transactions were used.
- ▶ Two TPNS networks were used, one to feed each of the IMS images. Each of the networks simulated between 200 and 300 terminals.
- ▶ Either 20 or 25 message regions were active.
- ▶ The IMS shared queues structures and logger structures were placed on the same Coupling Facility, which was local to both z/OS systems.

## 4.5.7 Utilities and tools

We used several tools to measure different parts of the system, and in many cases were able to cross check results from more than one tool in order to validate those results. The tools used were:

- ▶ RMF™ - for CPU, DASD, Workload, and CF activity
- ▶ IMSPA - for transaction and IMS statistical data from the IMS log
- ▶ IMS Monitor - with IMSPA to format
- ▶ IXGRPT1 - to format and print specific MVS logger statistics from SMF 88 records

### Sizing the IMS shared queues log stream

Approximately 80–90% of the log volume generated is the logging of the input and output messages. To make estimates of the size of these messages you can use the IMS logs to determine the average size of a message (x'01' and x'03 for full function, x'5901' and x'5903' for fast path EMH). There are several utilities and tools to help analyze this data and estimate the average message size, including the IMS Statistical Analysis Utility (DFSISTS0), the Fast Path Log Analysis Utility (DFSULTA0), and the IMS Performance Analyzer (program number 5655-E15). If using IMSPA for this purpose, be sure to have PTF UK23217 applied to get accurate average message sizes if you are already using shared queues.

### ***Administrative Data Utility (IXCMIAPU)***

This system utility (see Example 4-6 on page 131) can be used to determine the current size of the log stream. Specifically, it identifies the LOGR policy parameters for each log stream and tells you how many offload data sets are in use for each log stream.

### ***SMF and RMF***

The System Logger writes Type 88 SMF records, which can be used to monitor how much data is being written to, and deleted from, the log streams. There is information about how much data is written to the structure as well as to the staging data sets and offload data sets. There is a sample report job in SYS1.SAMPLIB (IXGRPT1), which will format and print the information in these SMF records.

RMF records Coupling Facility structure activity in SMF Type 74, Subtype 4 records. The CF activity can be monitored in real time using RMF Monitor III, or the RMF SMF records can be used to create an RMF Postprocessor report.

## 4.5.8 Measurement methodology and reports

As anyone familiar with performance monitoring knows, it is very difficult to get exactly the same number from two different products that are monitoring the same thing. Part of this may be due to small differences in precisely what is being measured. Part may be due to differences in how they gather the information (sampling versus measuring, for example). And part may even be due to small differences in the interval being measured, especially when there are hundreds of transactions being processed every second. These differences sometimes make it difficult to get precise values for some of the numbers being gathered.

We tried to get the numbers as close as possible by coordinating the RMF intervals with the taking of an IMS checkpoint and the switching of the OLDS. Our sequence of operations for each individual measurement was as follows:

1. Switch SMF data sets (I SMF command).
2. Wait for RMF report interval (set at 1 minute - on the minute).
3. Switch OLDS.
4. Take an IMS simple Checkpoint.
5. Wait 10 minutes.
6. Take another IMS Checkpoint.
7. Switch OLDS.
8. Wait for the next RMF interval (slightly less than 1 minute).
9. Switch OLDS.
10. Take an IMS simple Checkpoint.
11. Turn on the DC Monitor.
12. Wait 3 minutes.
13. Turn off the DC Monitor.
14. Take another IMS Checkpoint.
15. Switch OLDS.
16. Switch SMF again.

This procedure allowed us to keep very comparable data for each set of measurements. While the DC monitor was not absolutely necessary for this test, it helped us cross check our results.

## 4.5.9 Measurement results

The following results are averages of several measurements taken within the same environment to ensure consistency. All of the information in these tables was obtained from various RMF reports.

The first measurement in Table 4-1 is with the logger structure being duplexed to a data space. This was used as our base measurement to establish what could be achieved within our hardware, software, and application environment.

*Table 4-1 Log stream with duplexing to data space*

Metric	IM1A	IM2A	Total/average
Total z/OS % CPU usage	75.4	76.6	76 (avg.)
Tran. Rate/sec.	970	1002	1972 (total)
Tran. Resp. (ms)	11	10	11 (avg.)
Logger Str. Rate/sec	4832	4939	9770 (total)
Logger Str. Resp. time (us)	137	136	136 (avg.)

One thing that we would like to stress as you review these results is that you should always avoid using percentages when describing overhead. Describing overheads in percentage terms can be very misleading, especially when trying to apply them to a very different environment. If you think back to the description of our test application, you will remember that it performed no database calls. In this case, our application execution time will be a very small number, so any increases to that base time, when expressed as a percentage, will be quite large. On the other hand, if the application performed many complex SQL queries, but saw the same absolute change in response time, the much larger base execution time would have yielded much smaller percentage increases.

Before moving on to the next measurement, did you notice that the number of logger structure requests per transaction is almost 5 (9770/1792), not over 4 as was previously stated? This is because of the additional CF requests required to offload the log data as the structure reaches the predefined full threshold.

The results of the next test, shown in Table 4-2, are with duplexing the logger structure to a single staging data set that was being PPRCed over 20 km.

*Table 4-2 Log stream duplexing to a single staging data set*

<b>Metric</b>	<b>IM1A</b>	<b>IM2A</b>	<b>Total/average</b>
Total z/OS % CPU usage	25.9	27.8	26.85 (avg.)
Tran. Rate/sec.	292	320	612 (total)
Tran. Resp. (ms)	44	40	42 (avg.)
Logger Str. Rate/sec	1246	1392	2638 (total)
Logger Str. Resp. time (us)	169	167	168 (avg.)
Staging Data Set I/O/sec.	1181	1381	2562 (total)
Staging Data Set Resp Time (ms)	2.1	1.9	2.0 (avg.)

The IBM DS8K was able to sustain the rates and response times shown. Note that the DASD had the Parallel Access Volume (PAV) feature enabled, allowing more than one I/O at a time to be started to the volume.

You can see that the transaction throughput for each IMS image was limited by the I/O to the staging data set, and that the transaction response time has increased due to the additional time taken to write to the staging data sets. Given that each transaction results in roughly four IXGWRITE requests, and that the staging data set response time was about 2 ms, you would expect the transaction response time to increase to roughly 20 ms. (11 ms. without staging data set, plus 4 writes at 2 ms. each). In fact, the response time increased to about 42 ms. The difference is a result of additional queue time in IMS that resulted from the increased time it was taking each transaction to get in and out of IMS.

The next test, shown in Table 4-3, was done with the staging data sets using two stripes. You can see that the I/O rates and response times for each volume were lower than with the single staging data set, and the total I/O and transaction rates were higher. However, the transaction rates did not double as a result of having two stripes, as one might have hoped.

*Table 4-3 Log Stream duplexing with 2-stripe staging data sets*

<b>Metric</b>	<b>IM1A</b>	<b>IM2A</b>	<b>Total/average</b>
Total z/OS % CPU usage	31.7	30.9	31.3 (avg.)

Metric	IM1A	IM2A	Total/average
Tran. Rate/sec.	363	366	729 (total)
Tran. Resp. (ms)	35	34	34.5 (avg.)
Logger Str. Rate/sec	1564	1574	3138 (total)
Logger Str. Resp. time(us)	163	162	162 (avg.)
Staging Data Set I/O/sec.	692/687	716/710	2805 (total)
Staging Data Set Resp Time (ms)	1.5/1.3	1.5/1.2	1.4 (avg.)

The results shown in Table 4-4 are with four stripes for the staging data sets. As you can see, there was no additional improvement in throughput, although the individual data sets had less activity. This would infer that, for this particular workload, IMS was only driving two IXGWRITE requests concurrently, and therefore did not benefit from the ability to drive more than two concurrent writes to the staging data set. The use of striping for VSAM data sets is discussed in *VSAM Demystified*, SG24-6105.

Table 4-4 Log stream duplexing with 4-stripe staging data sets

Metric	IM1A	IM2A	Total/average
Total z/OS % CPU usage	30.1	29.7	29.9 (avg.)
Tran. Rate/sec.	350	350	700 (total)
Tran. Resp. (ms)	36	36	36 (avg.)
Logger Str. Rate/sec	1532	1530	3062 (total)
Logger Str. Resp. time(us)	164	161	162 (avg.)
Staging Data Set I/O/sec.	360/354/354/354	355/355/360/355	2847 (total)
Staging Data Set Resp Time (ms)	1.2/1.4/1.4/1.4	1.4/1.2/1.5/1.3	1.35 (avg.)

The next measurement was to keep the four-stripe staging data sets, but to turn off the PPRC mirroring to understand the effects of using staging data sets but with no distance involved. Another way to look at this is to see the effect of the 20 km distance on the staging data sets. Table 4-5 shows these results. As you will see, the total transaction throughput increased, while transaction response time decreased. This can be mainly attributed to the lower DASD response time to the staging data sets without the 20 km mirroring.

Table 4-5 Log stream duplexing to four-stripe staging data sets but NO PPRC

Metric	IM1A	IM2A	Total/average
Total z/OS % CPU usage	46.0	47.1	46.6 (avg.)
Tran. Rate/sec.	559	581	1140 (total)
Tran. Resp. (ms)	20	20	20 (avg.)
Logger Str. Rate/sec	2466	2607	5073 (total)
Logger Str. Resp. time(us)	144	152	148 (avg.)
Staging Data Set I/O/sec.	573/566/566/566	590/590/600/590	4641 (total)
Staging Data Set Resp Time (ms)	.5/.6/.6/.5	.6/.4/.6/.6	.55 (avg.)

In Table 4-6 we see what would happen if, instead of using staging data sets, we used system-managed duplexing for the logger structure at a distance of 20 km. Our total transaction volume decreased from 700 with the four-stripe staging data sets (mirrored) to 482 with duplexing the logger structure at the same distance. The structure response times increased drastically, causing overall throughput to decrease and transaction response times to increase. This indicates that modern, well-performing DASD may deliver better performance than the use of system-managed duplexing for system logger structures. This phenomenon is related to the combination of how system logger uses CF structures, and the multiple *handshakes* that are required for a system-managed duplexed request.

Table 4-6 Log stream with CF Level 14 system-managed duplexing for the logger structure

Metric	IM1A	IM2A	Total/average
Total z/OS % CPU usage	26.2	24.4	25.3 (avg.)
Tran. Rate/sec.	253	229	482 (total)
Tran. Resp. (ms)	52	58	55 (avg.)
Logger Str. Rate/sec	1241(p)/1191(s)	1411(p)/1314(s)	2512(p)/2570(s)
Logger Str. Resp. time(us)	2540(p)/2608(s)	2485(p)/2533(s)	2619 (avg.)

One final measurement was to see what would happen if we duplexed the SMQ structure instead of the logger structure (thereby hopefully eliminating the need to ever do a structure recovery). Table 4-7 shows the results of this configuration.

Table 4-7 System-managed duplexing for the SMQ structure

	IM1A	IM2A	Total/average
Total z/OS % CPU usage	27.2	26.3	26.75 (avg.)
Tran. Rate/sec.	267	268	535 (total)
Tran. Resp. (ms)	39	39	39 (avg.)
Logger Str. Rate/sec	1174	1175	2349 (total)
Logger Str. Resp. time(us)	148	140	144 (avg.)
MSGP Str. Rate/sec.	2631(p)/2296(s)	2640(p)/2305(s)	5271(p)/4601(s)
MSGP Str. Resp. (us)	572(p)/808(s)	573(p)/808(s)	573(p)/808(s)

While the SMQ structure response times did not show the drastic increases that the logger structure suffered, there are more structure accesses per transaction, so there was not as significant of an improvement as one might have expected from just looking at the structure response times.

## 4.5.10 Summary of measurement results

There are a lot of numbers in the above tables and, as with any measurements, they only show relative results for the environment in which they were done. Your message sizes and hardware configuration will almost certainly be different. However, the point was to try to show how CQS might react to different methods of duplexing and therefore allow you to have some idea of what to expect in your own environment. Here are some summary points from the measurements that were done:

- ▶ Duplexing the logger structure to a data space is by far the best option for performance. However, this may not be a valid option for certain configurations such as ICFs, volatile CFs, and when using DASD mirroring.
- ▶ Duplexing to staging data sets on the IBM DS8K makes the use of staging data sets a much more viable option than with earlier DASD subsystems. The DS8K was able to sustain very high I/O rates while maintaining good response times, even when mirroring at a distance of 20 km.
- ▶ Enabling striping for the staging data sets can give some throughput improvement over a single stripe. Perhaps more importantly is that striping can lower the I/O rate per volume, enabling improved DASD response times.
- ▶ System-managed duplexing of the logger structure is probably not as good an option as using staging data sets.
- ▶ System-managed duplexing of the shared queues message structure, while not having as severe an impact to response time as the logger structure, can still be a limiting factor to throughput.
- ▶ If staging data sets are implemented, then additional IMS images may be required in some cases to provide the necessary transaction throughput.

So, finally, there are several options available depending on your configuration and availability needs. While these measurements give some idea of what might be the impact of different processing options, there is no substitute for testing in your own environment with your own applications.

## 4.5.11 Additional considerations

Here are additional considerations

### SMF 88 fields of interest

The System Logger will generate SMF type 88 log records if requested in the SMFPRMxx member in SYS1.PARMLIB. These log records can be gathered and printed using a sample program and JCL found in SYS1.SAMPLIG (IXGRPT1 or IXGRPT1J). A sample output listing, and a description of the output fields, can be found in 8.6.4, “IXGRPT1 Field Summary and IXGSMF88 Cross Reference” on page 293.

For CQS, or any log writer, the first goal is to not let the logger structure fill up. This means that offload performance must be adequate to relieve structure constraints before they become a problem. The report (RPT1) shows different types of writes, shows structure full conditions, and average buffer size. All these numbers are for the SMF monitor interval only - they are not cumulative, so they may change significantly as different workloads become prevalent.

- ▶ A type 1 write means that CQS wrote a log record while the structure was below its high threshold. These should be the most common.



- ▶ A type 2 write means that CQS wrote a log record while the structure was at or above the high threshold. This is common and doesn't indicate any problems. An offload should already be in progress - probably just started.
- ▶ A type 3 write indicates that CQS wrote a log record while the structure was more than 90% full. This should *not* be common and may indicate that your HIGHOFFLOAD value is too high. You are close to filling up the structure. This may be a problem with the offload process not performing well enough to free up space. Data set allocation can sometimes slow it down. If you are getting many type 3s, lower the high threshold and increase the LS\_SIZE (fewer allocations).
- ▶ Structure full is bad. This means that CQS tried to write a log record but couldn't because there were no more entries or data elements in the structure. The write will fail and CQS will put out message CQS0350W. These are just an extension of the type 3s and you should take the same actions.
- ▶ Average buffer size is the total number of bytes written by IXGWRITE users divided by the number of writes invoked. This is the number used by the System Logger to recalculate the entry-to-element ratio, although the System Logger uses a 30 minute interval.

The SMF 88 records do not show the second important goal — not letting the log stream fill up (that is, all the offload data sets *and* the logger structure are full). This occurs when there are no more offload data sets available. This can be tracked using IXCMIAPU as described in 4.3.4, “Monitoring for offload conditions” on page 131.

### Sizing log streams

A lot has already been said about sizing the log streams and logger structures. The main criteria here is to not run out of offload data set space. You always need enough room on DASD to hold all the log data that is generated through *two* structure checkpoint intervals. The size of the logger structure is not as critical. A smaller structure just invokes the offload process more often. A good target is to make the structure large enough so that, whatever your HIGHOFFLOAD threshold is, offload processing will not be invoked more frequently than about once every 30-60 seconds during peak transaction processing times.

### Threshold recommendations

The only thresholds of interest to CQS are the HIGHOFFLOAD and LOWOFFLOAD thresholds in the log stream definition in the LOGR policy. Since CQS rarely needs to read these logs after they are written, the LOWOFFLOAD threshold should always be set to zero (0%) — that is, when offload processing is invoked, always offload as much as possible. The HIGHOFFLOAD threshold should be set between 50 and 80 (%). We recommend 50% since this leaves plenty of room in the structure in case offload processing gets delayed.

Archived



## CICS and System Logger

This chapter discusses how CICS Transaction Server (CICS) works with System Logger and how to manage the interaction between these components to deliver the required levels of availability and performance.

This chapter provides:

- ▶ Description of how CICS uses System Logger
- ▶ Definitions that must be set up to enable CICS use of System Logger
- ▶ How to size CICS log streams
- ▶ Configuration and performance considerations

## 5.1 Function description

CICS started using System Logger in the first release of CICS Transaction Server. The services of the System Logger are used to record information required for transaction back out and recovery on the CICS system log. The CICS system log is comprised of two log streams (*DFHLOG* and *DFHSHUNT*). In addition, data required for forward recovery may be written to a forward recovery log stream, and application data may be written to a user journal which is also stored in a log stream.

The period between the start of a particular set of changes and the point at which they are complete is called a unit of work (UOW). A UOW is a sequence of actions which must complete before any of the individual actions can be regarded as complete. A UOW is terminated by a sync point. The sync point may be issued implicitly at task end, or explicitly when the task issues an EXEC CICS SYNCPOINT. In the absence of user sync points being explicitly issued within the transaction, the entire transaction is one UOW. After changes are committed (by successful completion of the UOW and recording of the sync point on the system log), they become durable, and are not backed out in the event of a subsequent failure of the task or system. If a transaction consisting of multiple UOWs fails, or the CICS region fails, committed UOWs are not backed out.

The log records on the system log stream that are associated with a given UOW are “chained” together. In the context of the System Logger function, the term chain refers to the grouping of records for a particular UOW.

The interface to the System Logger is via the CICS Log Manager domain, referred to as the logger domain. CICS resource managers, such as file control, communicate with the recovery manager (RM) component of CICS. The recovery manager associates the request with the unit of work (UOW) and passes the request to the logger domain. The logger domain places the data into a buffer and then calls the System Logger using an IXGWRITE call to pass the data to be written to the log stream. When the UOW completes (that is, issues a sync point), CICS can then delete all log records associated with the UOW.

An application may also request user-defined data to be recorded in a user journal using an EXEC CICS WRITE JOURNALNAME command. The application execution interface will pass the request to the logger domain, where an IXGWRITE call is issued for the log stream associated with the journal name.

In order for CICS to connect to a log stream, the log stream must be defined in the System Logger policy. From a CICS point of view, the System Logger policy is a resource repository much like the CICS System Definition (CSD) file.

The log stream definition must exist in the System Logger policy before CICS can connect (think of a connect as an OPEN) and use it. If the log stream resources have not been predefined, CICS will attempt to dynamically define them as needed.

CICS provides support for the dynamic installation of many of its resources. Following in that tradition, log streams can be dynamically defined to the System Logger. To facilitate the dynamic definition of log streams, the names and characteristics must be supplied via models.

In the CICS System Logger environment, two types of models may be required.

The first is a journal model defined in CICS which provides a linkage between the journal name and the log stream to be used. The STREAMNAME parm in the JOURNALMODEL identifies the log stream to be used by CICS. The CICS journal models replace the journal control table used in CICS/ESA V4.1 and earlier.

For DFHLOG and DFHSHUNT, if the JOURNALMODEL is not defined, CICS will attempt to connect to a log stream named *&userid..&applid..DFHLOG* (or *&userid..&applid..DFHSHUNT*).

For non-system logs (user journals, forward recovery logs, and auto journals), if a JOURNALMODEL has not been defined, the name used would be *&userid..&applid..DFHJ0xx*, where xx is the journal number.

The second type is referred to as an MVS model. MVS log stream models are defined in the System Logger policy using the IXCMIAPU utility. The model contains the log stream characteristics to be assigned when the log stream is dynamically created. MVS models *sysname.DFHLOG.MODEL* and *sysname.DFHSHUNT.MODEL* are required entries in the System Logger policy if CICS is to dynamically create the DFHLOG and DFHSHUNT log streams. “*sysname*” is the name of the z/OS image as specified in the IEASYSxx in SYS1.PARMLIB. Figure 5-1 shows how CICS decides on the name of the log stream to be used for DFHLOG (as an example).

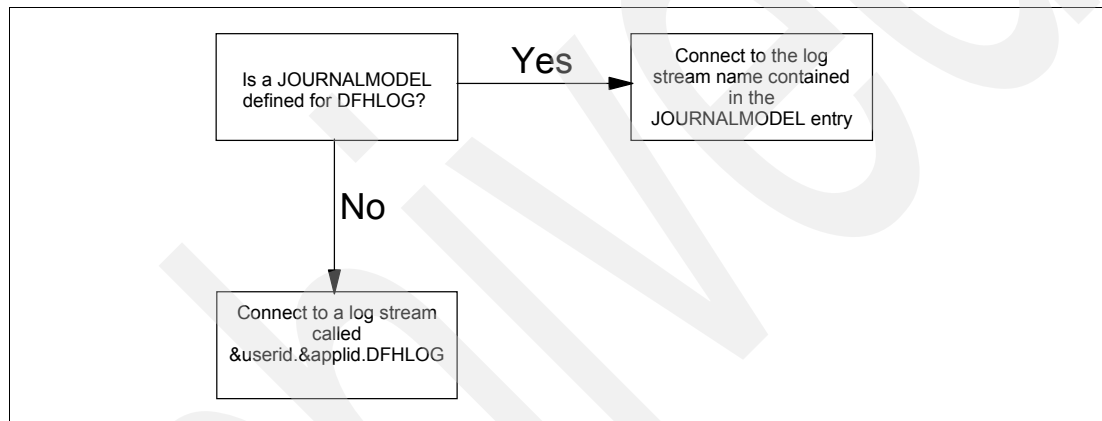


Figure 5-1 How CICS decides which log stream name to use

When CICS requests a log stream to be dynamically created for DFHLOG or DFHSHUNT, an IXGINVNT DEFINE request will be issued, pointing to an MVS model named *sysname.DFHLOG.MODEL* or *sysname.DFHSHUNT.MODEL* respectively. The name of the MVS model used for the log stream create may be changed in the CICS XLGSTRM exit. A sample exit is provided with CICS Transaction Server V2.2 in member DFH\$LGSL of SDFHSAMP. For earlier releases, a sample exit is available via a ServerPac which can be downloaded from:

<http://www.ibm.com/software/ts/cics/txppacs/cs1e.html>

All characteristics of the log stream to be defined may be changed in the XLGSTRM exit *except* the log stream name. Prior to calling the exit, CICS will have already cataloged the Log Stream Name in the CICS catalog..

**Tip:** All log streams defined using a given MVS model will have the same attributes, unless altered in the XLGSTRM exit. Having the same attributes for all log streams could result in incorrect definitions for the type and volume of data written, or it could result in all log streams being placed in the same structure.

The log stream names for DFHLOG and DFHSHUNT can only be changed when CICS is *initial* started. For other types of starts (cold, warm, and emergency) the log stream name is obtained from the CICS catalog.

During CICS startup, an IXGCONN call will be issued to connect to the DFHLOG and DFHSHUNT log streams. CICS will connect to user journals and forward recovery logs at first reference.

Note that CICS allocates two buffers for each log stream: this enables CICS to write while an IXGWRITE is in progress for the alternate buffer.

During a normal shutdown, CICS will flush the log buffers and issue a disconnect call (IXGDISC) for the log streams. However, it's important to note during an immediate shutdown (CEMT P SHUT,I) the buffers are *not flushed* for user journals.

To avoid loss of user journal records, use a normal shutdown with the shutdown assist program (DFHCESD) active. Add a program to the Program List Table for ShutDown (PLTSD) to issue a SET JOURNAL FLUSH for each user journal. Even though the shutdown assist program may force an immediate shutdown, it will happen after the PLTSD programs have run.

Another method is to add the wait option on writes to user journals; however, use of the wait option could lead to performance and response time issues.

### 5.1.1 Log stream usage

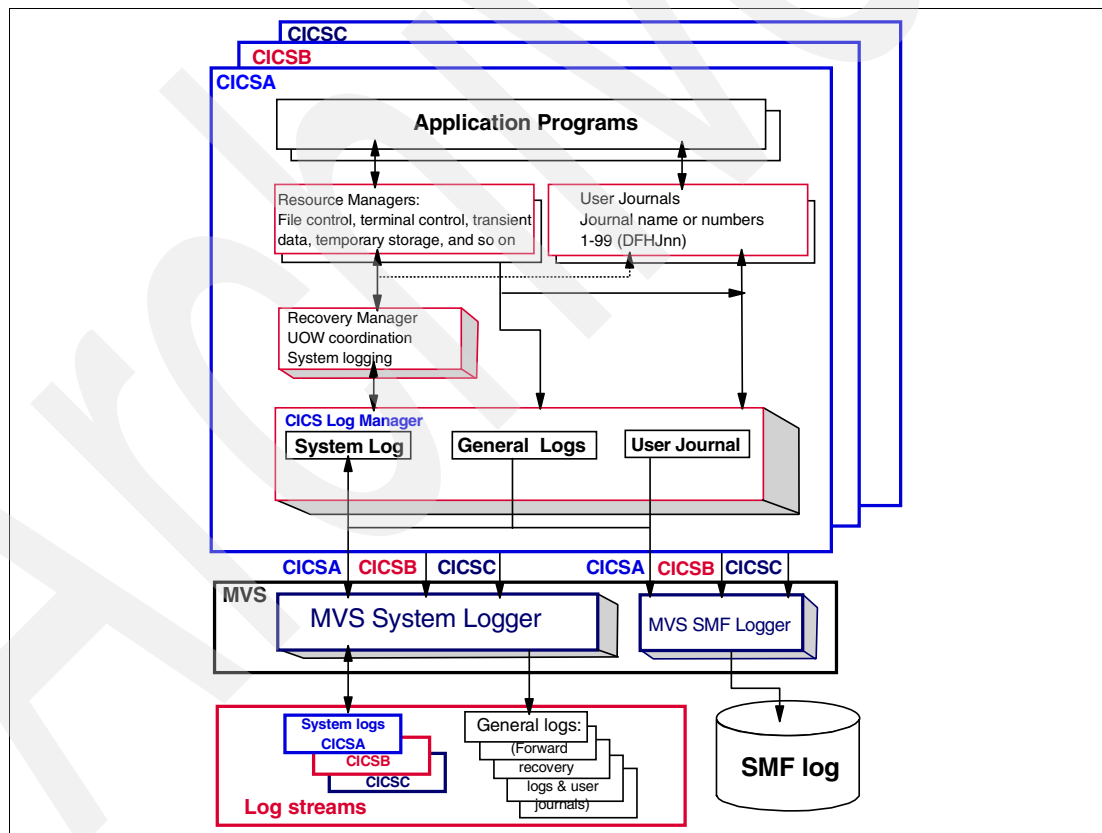


Figure 5-2 Overview of CICS logging

Figure 5-2 shows the relationship of the resource managers, the logger domain, and the System Logger function.

Log streams used by the CICS address space can be categorized into four basic types:

- ▶ System
- ▶ Forward recovery
- ▶ User
- ▶ Autojournal

In addition, the log of logs is a special form of a user journal.

- ▶ The system log is comprised of two log streams, DFHLOG and DFHSHUNT.

The system log is used to hold information required to perform backout of in-flight activity in the event of a transaction, region, or system failure.

Recovery information, such as the “before” image of the data when an update is made to a recoverable VSAM file, is written to DFHLOG prior to the update. A transaction does not cause a log record to be written until it accesses a recoverable resource for update. There are many tasks, such as CEMT which does not update recoverable resources and therefore does not cause log records to be written.

DFHSHUNT is a dual purpose log stream. In the case where a task has caused a log record to be written to DFHLOG and it does not cause another record to be written to the log within one activity keypoint, its records are moved to DFHSHUNT. This means the chain of log records (known as a log chain) written on behalf of the task since the last sync point, or start of task, is moved to DFHSHUNT. By moving the records for tasks which are inactive for one activity keypoint to DFHSHUNT, CICS is better able to manage the tail (oldest records required for backout) on DFHLOG.

The second use of DFHSHUNT is to hold the log chain for a UOW which is in INDOUBT status. A UOW is INDOUBT in the resource manager region if it has been invoked for the first phase of sync point (prepare), but not for the second (commit). A failure in the resource manager, CICS, or the communication link in the period between Phase 1 and Phase 2 leaves the resource manager in doubt as to whether the UOW should commit or backout. In those cases the log chain associated with the UOW is moved from DFHLOG to DFHSHUNT to await resynchronization.

The only records written to DFHSHUNT are those which have been moved from DFHLOG for one of these two reasons.

When a transaction completes, or issues a user SYNCPOINT, the records for the UOW have been committed and are no longer required for backout.

Log tail management is the process of removing records from the DFHLOG and DFHSHUNT log streams which are no longer required. The oldest record of interest to CICS on the log stream is defined as the *history point*. During the activity keypoint process, CICS will determine the history point for DFHLOG and DFHSHUNT. An IXGDELET call will be issued to inform the System Logger all records older than the history point may be logically deleted. The records are physically deleted during the next offload process for that log stream.

Performance of DFHLOG has a major impact on transaction throughput and response time. For a description of CICS parameters which can affect log stream performance, refer to “AKPFREQ” on page 153 and “LGDFINT” on page 154.

Log tail management is only performed for DFHLOG and DFHSHUNT.

- ▶ Forward recovery log

CICS supports forward recovery of VSAM data sets updated by CICS file control. If a forward recovery log is defined in either the file definition or the ICF catalog entry for the data set, the “after” image of changes will be logged.

Data written to the forward recovery log is appended to the exiting data in the buffers for the log stream. The buffers are written (an IXGWRITE issued) to the log stream during sync point processing.

CICS itself does not use the data from a forward recovery log. The data is used by products designed to provide forward recovery, such as IBM CICS/VR.

Log tail management is not performed on forward recovery logs.

- ▶ User journals

User journals contain data written as a result of an EXEC CICS WRITE JOURNALNAME or EXEC CICS WRITE JOURNALNUM command from an application program.

If the wait option is specified on the write command, the log write (IXGWRITE) will be issued immediately to cause the buffer to be written to the log stream. Otherwise, the buffer is written when it fills.

Data written to a user journal is not synchronized with the UOW.

CICS does not reference the data written to user journals.

Log tail management is not performed on user journals. The program used to process the log stream could delete the data using IXGDELETE calls, or you could define the log stream with a RETPD>0 and AUTODELETE(Y).

- ▶ Auto journal

CICS supports the use of an auto journal for file control data and terminal control messages. Auto journals are generally used for audit trails.

When a file is defined to CICS, auto journaling may be specified. Data for the operations selected is recorded on the journal specified in the CICS file definitions. For example, JNLUPDATE(YES) indicates rewrite and delete operations are to be recorded on the log stream.

For operations defined as synchronous (JNLSYNCREAD or JNLSYNCWRITE) the log write operation is issued immediately, otherwise the buffer is written when it fills.

Auto Journal of terminal messages is defined in the communications profile.

The data written to an auto journal is not referenced by CICS. You must provide your own programs to process the data.

Log tail management is not performed on auto journals. The program used to process the data could delete the data using IXGDELETE calls, or the log stream could be defined with a RETPD>0 and AUTODELETE(Y).

- ▶ Log of Logs

The log of logs is a form of a user journal written to provide information to forward recovery programs such as CICS VSAM Recovery (CICS/VR). The log contains copies of the tie-up records, which provide a summary of VSAM recoverable data sets used and the log stream to which the forward recovery information was written.

CICS does not perform log tail management on the log of logs log stream. The program used for forward recovery would normally be used to delete the data using IXGDELETE calls, or the log stream could be defined with a RETPD>0 and AUTODELETE(Y).

## 5.1.2 Criticality/persistence of data

The data written to DFHLOG and DFHSHUNT persists only for the time period it might be required for recovery, in the event of a transaction, region, or system failure. For general logs (forward recovery, auto journals, or user logs), the data persists for the duration of the



retention period specified in the log stream definition, or until the data has been deleted. Refer to Figure 5-15 on page 173 for information concerning offload data set deletion.

## 5.2 Definitions

To implement the System Logger in a CICS environment, a number of definitions are required as shown in Figure 5-3. The figure is organized by task on the left with the related policy listed on the right. Each of the z/OS policies can be considered similar in concept to the CICS System Definition (CSD) file.

The utility used to define resources in the System Logger and CFRM policies is IXCMIAPU, which is similar in function to DFHCSDUP for CICS.

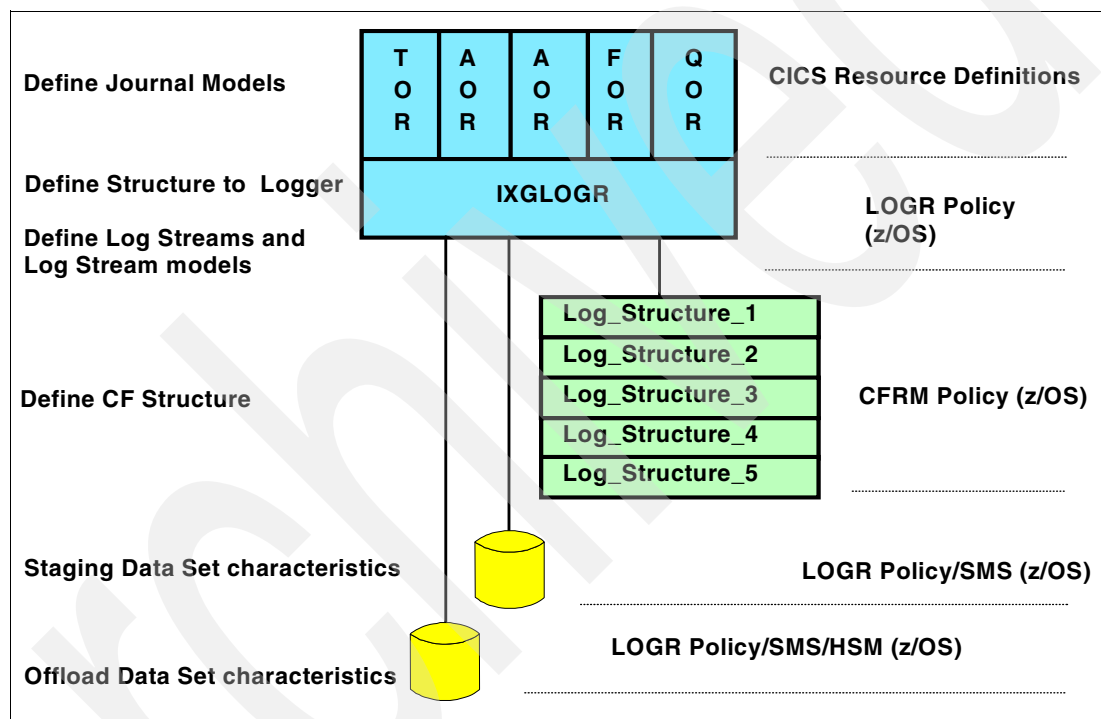


Figure 5-3 System Logger-related definitions

### 5.2.1 Subsystem definitions

As shown in Figure 5-3, there are many parts which must be in place prior to a log stream being used by CICS. The log streams can be predefined, or dynamically installed. In either case, if a specific log stream name (other than the default generated by CICS) is to be used when CICS connects to the log stream, it must be supplied in a CICS journal model.

When a CICS region is started, it connects to the DFHLOG and DFHSHUNT log streams which make up the system log. If journal models have been defined for DFHLOG and DFHSHUNT, CICS uses the log stream name (LSN) found in the model for the connect.

If a JOURNALMODEL has not been defined, &USERID.&APPLID.DFHLOG (or DFHSHUNT) is used as the LSN.

For non-system logs, CICS searches for a journal model, for example DFHJ07 for user journal 7. If a journal model is found, the LSN provided is used. If a journal model is not found,

a log stream name of &userid.&applid.&jnam, for example GRAUEL.IYOT1.DFHJ07, is used.

### **CICS Journal Models**

Figure 5-4 provides a sample job for defining a CICS journal model. The important parameters are JOURNALNAME, TYPE, and STREAMNAME. The JOURNALNAME is the name used in the CICS environment such as DFHLOG, or DFHJ07. The TYPE must be "MVS" and streamname identifies the log stream name (LSN).

### **JOURNALMODEL**

The JOURNALMODEL may contain an explicit log stream name (for example, JIMG.JIMSAOR.DFHLOG) or a TEMPLATE value (for example, &applid..DFHLOG), in which case the symbolic qualifiers are used to build the actual log stream name.

```
//DFHCSDUP JOB CLASS=A,MSGCLASS=A
//UPGRCSDD EXEC PGM=DFHCSDUP
//STEPLIB DD DSN=CICSTS22.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE JOURNALMODEL(CICSLOG) The name of the model is CICSLOG
GROUP(TEST) The CICS group name is TEST
DESCRIPTION('DFHLOG LOGSTREAM USING A TEMPLATE DEFINITION')
JOURNALNAME(DFHLOG)
TYPE(MVS) Records to be written to the MVS log stream
specified.
STREAMNAME(&USERID..&APPLID..DFHLOG) The CICS logger issues an IXGCONN
to connect to the streamname derived from the
template or the name explicitly specified.

DEFINE JOURNALMODEL(SYSSHUNT)
TYPE(MVS) GROUP(TEST) JOURNALNAME(DFHSHUNT)
DESCRIPTION('DFHSHUNT LOGSTREAM EXPLICIT DEFINITION')
STREAMNAME(TCOM.IYCLZCCA.DFHSHUNT)
```

Figure 5-4 Sample CICS JOURNALMODEL

Valid symbolic qualifiers are:

- &USERID** The symbolic name for the CICS region user ID, which can be up to eight characters. If the region does not have a user ID, the string "CICS" will be used.
- &APPLID** The symbolic name for the CICS region APPLID, as specified in the APPLID system initialization parameter, and can be up to eight characters.
- &JNAM** The symbolic name for a journal name which references, either by a specific or generic match, this journal model definition. &JNAME can be up to eight characters in length.
- &SYSID** The symbolic name for the CICS region SYSID as specified in the SYSIDNT system initialization parameter. If SYSID is not specified, the string "CICS" will be used.

**Tip:** Log stream names (LSN) can only have a maximum of 26 characters. The reason for the restriction is in the offload (log) data set names. Offload data sets contain the LSN with a prefix of the High Level Qualifier (HLQ) as specified on the System Logger policy definition, plus an 8-character sequence number.

For example, if the Log Stream Name (LSN) is IYOT1.DFHLOG, and the HLQ is CICS\_LVL2, the offload data set name would be CICS\_LVL2.IYOT1.DFHLOG.Axxxxxxx

When the log stream is defined, the first offload data set will be allocated. If the LSN is greater than 26 characters and the HLQ is 8 characters (the sequence number will always be 8 characters), the allocation will fail.

The failure is fairly easy to spot when the log stream is being defined with IXCMIAPU. When the log stream is being dynamically defined during a CICS initial start, the failure is not so easy to spot. CICS will terminate with a DFHLGxxx message with an IXC2511 message is written to the console.

Once the log stream name is defined, CICS will issue an IXGCONN request to connect to the log stream. If the log stream has not been defined in the System Logger policy, the System Logger returns IXGRSNCODENOSTREAM (RC 8 reason code 080B).

During a CICS *initial* start, if the log stream for DFHLOG (or DFHSHUNT) is not found, upon receiving an 080B, CICS attempts to create the log stream via an IXGINVNT DEFINE request using the MVS model *sysname.DFHLOG.MODEL*. The model used for DFHSHUNT is *sysname.DFHSHUNT.MODEL*. In all cases the log stream name remains the same — the one used for the original IXGCONN call.

For a non-system log (user journal, forward recovery log, or auto journal), the name of the logger model used consists of the first two qualifiers of the log stream name with MODEL appended (for example, if the log stream name is JIM.JON.HARRY - the model name will be JIM.JON.MODEL). The MVS models must exist in the System Logger policy (that is, a DEFINE LOGSTREAM has been issued for a log stream called JIM.JON.MODEL with MODEL(YES) specified).

If the create request for either DFHLOG or DFHSHUNT fails, CICS will abend, with a message in the range of DFHLG0503 through 0511 being issued.

## **DFHSIT Parameters**

The CICS System Initialization parameters which affect the interaction with the System Logger are AKPFREQ (Activity Key Point Frequency) and LGDFINT (Log Defer Interval).

### ***AKPFREQ***

The activity keypoint frequency value (AKPFREQ) specifies the number of times CICS appends log records to the buffer associated with the DFHLOG log stream before an activity keypoint is initiated. It does not reflect how much data is held in the buffer prior to it being written to the log stream. The data is written as required (based on the buffer filling, or the force option being specified on the call to the CICS logger domain). A keypoint is a snapshot of in-flight tasks in the system at the time. The activity keypoint interval is defined as the interval of time between two consecutive activity keypoints.

During the activity keypoint, CICS initiates the tail trimming process for DFHLOG and DFHSHUNT. CICS determines the history point (the oldest record required for transaction backout) for DFHLOG and DFHSHUNT and issue an IXGDELET call to the System Logger to logically delete blocks of data older than the history point. Logically deleted data will be

discarded by the System Logger during the next offload process for the log stream. It's important to note the oldest record required defines the new tail of the log. No data is deleted which is newer than the history point.

CEMT I SYS may be used to view and dynamically change the value for AKPFREQ.

### **LGDFINT**

LGDFINT specifies the “log defer” interval. The log defer interval is the length of time CICS delays a forced journal write (a write which is considered immediate) before issuing the IXGWRITE call. The intent allows additional records to be placed in the buffer and reduce the number of IXGWRITE calls to the System Logger.

The LGDFINT mechanism applies only to DFHLOG and DFHSHUNT log streams.

However, when a buffer fills it is written without waiting for the log defer interval.

When a journal write call is passed to the logger domain with the wait option, the record is simply appended to the current buffer. The buffer is written either when it fills and the application will wait for the completion of the IXGWRITE.

When a journal write call is passed to the logger domain with the force option, the logger domain is synchronously issuing an IXGWRITE and the record is immediately written to the log stream.

The LGDFINT default value in releases prior to CICS Transaction Server V2.2 was 30 milliseconds (ms). In V2.2 the default was changed to 5 ms. The recommendation is LGDFINT be set to 5 for all releases.

CEMT I SYS may be used to view and dynamically change the value for LGDFINT.

## **5.2.2 System Logger definitions**

As shown in Figure 5-3 on page 151, most of the actual set up is for resources required from the z/OS point of view.

CICS provides several sample jobs (DFHILG1-7) in the SDFHINST library to assist with log stream definitions. However, the jobs *do not* contain installation-specific information. The jobs should be used as JCL examples. The following samples are provided to show parameters which affect performance and suggested settings in a CICS environment. However, note that the sizes shown *must be replaced* with values which match the needs of the log stream being defined.

### **Log streams**

CICS can use CF-Structure log streams or DASD-only log streams. The definition and recommendations differ depending on the type being used. Definitions for a CF-Structure based log stream will be discussed first, and then the DASD-only log streams in “DASD-only log stream” on page 164.

### **CF-Structure log stream**

The following are sample JCL and IXCMIAPU statements to define CF-Structure log streams. The parameters and values used are discussed in each of the following sections.

## Define a Structure for CICS log streams in the CFRM Policy

Figure 5-5 illustrates defining the LOG\_JG structure in the CFRM policy.

```
//DEFSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

DATA TYPE(CFRM) REPORT(YES)
DEFINE STRUCTURE NAME(LOG_JG)          LOG_JG is the structure name
    INITSIZE(20000)                    INITIAL structure size, in 1K units
    SIZE(30000)                        Structure can be ALTERed up to this size
    PREFLIST(CF01,CF02)                Specifies CF preference selection order
    ALLOWAUTOALT(NO)                   XES can not automatically alter this structure
    FULLTHRESHOLD(90)                  Issues messages if structure reaches 90% full
```

Figure 5-5 Defining structure in CFRM policy

### **FULLTHRESHOLD**

FULLTHRESHOLD (added in OS/390 2.9) specifies a percentage value used by automatic alter processing and structure full monitoring. If the structure exceeds this percent utilization, auto alter may adjust the structure (if you specified ALLOWAUTOALT(YES)) and issue messages to the console. The value specified on this keyword should be larger than the HIGHOFFLOAD for the log streams in this structure.

If you disable structure full monitoring by specifying FULLTHRESHOLD(0), no messages are displayed and information from the SMF Type 88 records should be used to tune the log streams.

## ALLOWAUTOALT

If ALLOWAUTOALT(YES) (added in OS/390 R10) is specified, when the FULLTHRESHOLD value is reached, XCF will issue message IXC585E *and* automatically start an ALTER against the structure to relieve the storage shortage. XCF will issue message IXC588I to the system log to externalize the ALTER request. Refer to Figure 5-6 for an example of the XCF messages.

```
MV2C 01145 03:51:49.50      00000080 *IXC585E STRUCTURE LOG_JG2 IN COUPLING FACILITY SVCF04, 979
      979 00000080 PHYSICAL STRUCTURE VERSION B5E2748F D0885E02,
      979 00000080 IS AT OR ABOVE STRUCTURE FULL MONITORING THRESHOLD OF 80%.
      979 00000080 ENTRIES: IN-USE:      930 TOTAL:      1105, 84% FULL
      979 00000080 ELEMENTS: IN-USE:      2401 TOTAL:      8841, 27% FULL

MV2C 01145 03:51:49.52      00000280 IXC588I AUTOMATIC ALTER PROCESSING INITIATED 980
      980 00000080 FOR STRUCTURE LOG_JG2.
      980 00000080 CURRENT SIZE:      3072 K
      980 00000080 TARGET SIZE:      3072 K
      980 00000080 TARGET ENTRY TO ELEMENT RATIO:      1192 :      3078
      980 00000080 TARGET EMC STORAGE PERCENTAGE: 0.00

MV2C 01145 03:51:58.30      00000280 IXC590I AUTOMATIC ALTER PROCESSING FOR STRUCTURE LOG_JG2 982
      982 00000080 COMPLETED. TARGET ATTAINED.
      982 00000080 CURRENT SIZE:      3072 K TARGET:      3072 K
      982 00000080 CURRENT ENTRY COUNT:      2921 TARGET:      2921
      982 00000080 CURRENT ELEMENT COUNT:      7542 TARGET:      7542
      982 00000080 CURRENT EMC COUNT:      0 TARGET:      0

MV2C 01145 03:52:53.33      00000080 IXC586I STRUCTURE LOG_JG2 IN COUPLING FACILITY SVCF04, 998
      998 00000080 PHYSICAL STRUCTURE VERSION B5E2748F D0885E02,
      998 00000080 IS NOW BELOW STRUCTURE FULL MONITORING THRESHOLD.
```

Figure 5-6 Example of Auto Alter messages

Because the offload capability provided by System Logger is threshold driven, you do not want XES adjusting the structure size at the same time System Logger is trying to manage the log streams within the existing structure space. Similarly, because System Logger provides its own function for dynamically adjusting the entry-to-element ratios, you do not want XES making similar adjustments independently of System Logger. Therefore, you should always specify ALLOWAUTOALT(NO) for any System Logger structure.

Additional information can be found in the topic “Define the Coupling Facility Structures Attributes in the CFRM Policy Couple Data Set” in *z/OS MVS Setting Up a Sysplex*, SA22-7625.

### SIZE

SIZE(nnnn) Specifies the maximum amount of space to be allocated for the structure in the CF. The number is specified in units of 1K (1024 bytes). The maximum structure size is the largest size to which the structure can be dynamically altered. Specifying too large a maximum structure size can waste CF resources.

### INITSIZE

INITSIZE(nnnn) Specifies the initial amount of space to be allocated for the structure in the CF. The number is specified in units of 1K (1024 bytes). The INITSIZE value must be less than or equal to the SIZE value.

### MINSIZE

MINSIZE(nnnn) MINSIZE specifies the smallest size in units of 1K (1024 bytes) to which the structure can be altered.

## Define a Structure for CICS log streams in the System Logger Policy

If you plan to use CF-Structure log streams, the structure each log stream is associated with must be defined in the System Logger policy. In addition, it must also be defined in the CFRM policy. Figure 5-7 contains a sample job to define the LOG\_JG structure in the System Logger policy.

The structure used for the log stream can affect performance. Key factors include the number and characteristics of the log streams which share the structure. We recommend limiting the number of log streams which can reside in the structure (LOGSNUM specified on the structure definition) to ten. All log streams in a structure should have similar characteristics. For example, TORs, AORs, and FORs typically have different log stream record size and volumes, so their DFHLOG and DFHSHUNT log streams should typically be in different structures.

```
//DEFSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

DATA TYPE(LOGR) REPORT(YES)
DEFINE STRUCTURE NAME(LOG_JG)          LOG_JG is the structure name
LOGSNUM(10) *                          up to 10 logstreams can connect
AVGBUFSIZE(400) *                       size of the starting 'average' buffer - monitor
                                         the 'effective average buffer' using IXCMIAPU
MAXBUFSIZE(64000) *                     determines the size of each log buffer - also
                                         defines the CF element size
                                         > 65276 - element size is 512
                                         =<65276 - element size is 256

* parms which affect performance
```

Figure 5-7 Defining a structure for CF-Structure log streams in the System Logger policy

### SIZE

The size of a structure is specified in the CFRM policy. Each structure is divided into *entries* and *elements*. Each IXGWRITE uses one entry and one or more elements, depending on the length of data written.

The entries in the structure are shared between all connected log streams on an as-needed basis, while the elements are divided equally across the number of connected log streams in the structure. When another log stream connects to the structure, the elements are dynamically redistributed. Similarly, when a log stream disconnects from the structure, the elements previously allocated to that log stream are redistributed equally across all remaining log streams in the structure.

### MAXBUFSIZE

The MAXBUFSIZE value you specify is returned to CICS when it connects to a log stream in that structure. CICS sets up its internal log buffers to be the same size as MAXBUFSIZE. For user journals, where the application does not use the wait option, it may be advantageous to specify a smaller size, as the buffer only gets written to the log stream when it fills. As a general rule, we recommend a MAXBUFSIZE of 64000.

MAXBUFSIZE in conjunction with AVGBUFSIZE is used to determine the entry-to-element ratio for that structure. When data is written to the CF, it is written in increments equal to the element size. A MAXBUFSIZE greater than 65276 gives an element size of 512; a MAXBUFSIZE equal to or less than 65276 results in an element size of 256. For example, the definition would result in a CF element size of 512 bytes:

```
MAXBUFSIZE(65532) AVGBUFSIZE(1100)
```

Three elements will be used for an average write (1100/512 and round up). The entry-to-element ratio is therefore 1:3. Beginning with OS/390 1.3, System Logger will dynamically adjust the entry-to-element ratio.

System Logger samples the structure use prior to adjusting the entry-to-element ratio. If the log streams using a given structure have very dissimilar characteristics, the sample may not reflect the data which will be written in the next interval. This would lead System Logger to adjust the ratio to a value which could result in a shortage of entries. For example, imagine an FOR which normally writes 1000 800 byte records per second, and an AOR which averages 500 100 byte records per second are sharing the same structure. One entry is required for each IXGWRITE. Further assume the structure is allocated to hold 2000 800 byte writes, with a MAXBUFSIZE of 64000. If System Logger finds the FOR had dominated the last interval (normally about 30 minutes) it will bias the ratio towards the needs of the FOR—in this case 1:4 (800/256 and round up).

Now, if during the next interval, the AOR dominates, writing 5000 100 byte records, it is possible to exhaust the entry pool without hitting HIGHOFFLOAD.

When the total number of entries in the structure become 90% full, *all* log streams in the structure are completely offloaded (that is, offload processing is initiated and completes when LOWOFFLOAD threshold is reached)

The effective average buffer size (the current average amount of data written per IXGWRITE call), is included in an IXCMIAPU report, as shown in Figure 5-8. Note this is the *average* across all the log streams in the structure—some log streams may have far smaller log blocks and others may have larger ones. Ideally all the log streams sharing a given structure would have similar average log block sizes.

STRUCTURE NAME(LOG_JG_20M) LOGSNUM(10)	
MAXBUFSIZE(64000) AVGBUFSIZE(400)	
<u>EFFECTIVE AVERAGE BUFFER SIZE(1024)</u>	
log stream NAME	CONNECTION
-----	-----
IYOT3.DFHSUNT	NO
IYOT2.DFHSUNT	NO
log streams CURRENTLY DEFINED TO THIS STRUCTURE(2)	

Figure 5-8 Using IXCMIAPU report to get Effective Average Buffer Size

The current entry-to-element ratio can also be calculated from data available in the Coupling Facility Activity report produced from the SMF 74 records using the RMF Post Processor utility, ERBRMFPP. A sample Coupling Facility Activity Report is shown in Figure 8-2 on page 287.



## Define the log stream in the LOGR policy

Figure 5-9 shows sample JCL and IXCMIAPU statements to define a CF-Structure log stream. The parameters and values used are discussed in the following sections.

<code>//DEFSTR JOB CLASS=A,MSGCLASS=A</code>	
<code>//POLICY EXEC PGM=IXCMIAPU</code>	
<code>//SYSPRINT DD SYSOUT=A</code>	
<code>//SYSIN DD *</code>	
<code>DATA TYPE(LOGR) REPORT(YES)</code>	
<code>DEFINE LOGSTREAM NAME(IYOT1.DFHLOG)</code>	Matches name in CICS Journal Model
<code>AUTODELETE(NO) *</code>	Always NO for DFHLOG and DFHSHUNT
<code>DIAG(YES)</code>	If an 804 (IXGRSNCODENOBLOCK) reason code will be given, a dump of the logger is taken before returning to CICS
<code>EHLQ(NO_EHLQ)</code>	Allows specification of 33 characters for a high level qualifier - mutually exclusive with HLQ
<code>HIGHOFFLOAD(80) *</code>	Threshold, expressed as a percent of the logstream space, when offloading is to take place
<code>HLQ(GRAUEL)</code>	HLQ for the offload and staging data set names
<code>LOGGERDUPLEX(COND) *</code>	Specifies whether Logger will continue to provide its own log data duplexing, regardless of other duplexing (such as system-managed duplexing).
<code>LOWOFFLOAD(40) *</code>	Value, expressed as a % of the space used by the logstream, which defines the offload target
<code>LS_DATACLAS(LS10MEG) *</code>	SMS data class used for DASD offload
<code>LS_SIZE(500) *</code>	The number of 4K control intervals to be allocated for each offload data set
<code>MODEL(NO)</code>	
<code>OFFLOADRECALL(NO) *</code>	Always set to NO for CICS logstreams
<code>RETPD(0) *</code>	Retention period, zero for DFHLOG and DFHSHUNT
<code>STG_DUPLEX(YES) *</code>	
<code>DUPLEXMODE=(COND) *</code>	Log writes are to be duplexed to the Staging Data set if the CF becomes volatile or failure dependent (CF and z/OS in same CPC)
<code>STG_SIZE(9000)</code>	No. of 4K control intervals in Staging data set
<code>STRUCTNAME(LOG_JG)</code>	Structure which will contain the data
* indicates parameters that affect performance	

Figure 5-9 CF-Structure log stream definition

### AUTODELETE and RETPD

AUTODELETE and RETPD can have a disastrous effect on DFHLOG and DFHSHUNT if specified other than AUTODELETE(NO) and RETPD(0). With AUTODELETE(YES) and RETPD>0, even though CICS attempts log tail management, all data will be off-loaded to the offload data sets and held for the number of days specified for RETPD. AUTODELETE(YES) allows the System Logger (rather than CICS) to decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified with RETPD(0), the System Logger will delete the old offload data set (and the data). If CICS needs the data for backout, the result will be an 804 return code and CICS will terminate with a DFHLG0772.

For the forward recovery log stream and log of logs, the recommended values are AUTODELETE(NO) and RETPD(xx) where xx is the number of days the data needs to be kept before allowing any deletion. This value is based on installation business needs. If you are planning to keep this data for a long period, be prepared to manage offload data sets.

### **DIAG(YES)**

DIAG(YES) should always be specified for DFHLOG and DFHSHUNT. In a case where the System Logger is unable to locate data requested by CICS, a response of return code 8 with reason code IXGRSNCODENOBLOCK (804) is given. This means backout data is not available and CICS treats it as a fatal error, terminating the region with a DFHLG0772. In many cases, information from the System Logger address space is needed to resolve the problem. When DIAG(YES) is specified in the log stream definition, a dump of the System Logger address space is provided (by the System Logger) in addition to the DFHLG0772 dump provided by CICS. There is no overhead associated with this parameter unless a dump is requested. This error is normally seen when CICS issues an IXGBRWSE for backout data or during activity keypoint processing when CICS issues the IXGDELET command to trim the tail of the log. It can also be returned during region startup when an IXGCONN (connect) is issued to connect to the log stream.

### **HIGHOFFLOAD**

The HIGHOFFLOAD parameter, in conjunction with the size of the log stream structure, has a major impact on the amount of virtual storage used by the System Logger. For a CF-Structure log stream, if staging data sets are not being used, all the data in the CF portion of the log stream is duplexed to the System Logger-owned data space. If staging data sets *are* used with a CF-Structure log stream, the data is written to the staging data set rather than the data space.

If the HIGHOFFLOAD value is too high, the log stream may fill before offload processing can free sufficient space in the log stream. CICS is not aware the offload process is taking place, and will continue writing to the log stream. This can lead to an entry or structure full condition, which causes log writes to be suspended for 3 seconds.

We recommend setting HIGHOFFLOAD to 80–85% for all CICS log streams.

### **LOWOFFLOAD**

The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the CICS environment, the LOWOFFLOAD value should be high enough to retain the data required for backout but low enough to keep the number of offloads to a reasonable number.

LOWOFFLOAD should be set between 40 and 60% for DFHLOG and DFHSHUNT. It should be set to 0 for user journals, forward recovery logs, and auto journals.

In tests setting LOWOFFLOAD closer to HIGHOFFLOAD (for example, 75 with a HIGHOFFLOAD of 80) some improvement was noted in transaction throughput at a cost of an increase in the storage frames and CPU in the System Logger address space. The storage frames used will normally increase as the size of the structure increases. Each region should be evaluated based on its workload.

It should also be noted setting the LOWOFFLOAD to a higher value may mask poorly behaving transactions. It may appear the log stream is working properly (for example little or no data is offloaded) when in fact large amounts of storage are being consumed to retain records which would be offloaded with a lower setting. Remember the goal is to contain the data needed for recovery using a minimum of interim storage, with no data being moved to the offload data sets.

### **LS\_SIZE**

LS\_SIZE defines the allocation size, in 4 KB blocks, for the *offload* data sets. It should be specified large enough to contain several offloads, possibly up to a day's worth. DFHLOG and

DFHSHUNT should only offload a minimal amount of data, if any. For user journals, *all* data is offloaded.

If you have log streams which offload large volumes of data, find a balance between a size small enough so the system can always find free space, while at the same time, minimizing the number of times new offload data sets need to be allocated.

It's very important to specify LS\_SIZE large enough to limit the number of times an additional data set is allocated, to reduce the exposure to delays during the allocation process. The size to be specified is based on the amount of data written during the prime processing window of the day. The amount of data written can be determined using the SMF Type 88 records produced for the log stream by the System Logger.

If the extent size of the SMS data class specified on the LS\_DATACLAS (or the value specified in LS\_SIZE) is too small, frequent DASD shifts (allocation of a new offload data set) can occur. Frequent DASD shifts have a negative effect on performance and expose the system to a depletion of DASD space. The number of offload data sets is limited by the DSEXTENT value specified when you format the LOGR Couple Data Set. The DSEXTENT value defines the number of System Logger directory entries. Each directory entry can point to a maximum of 168 offload data sets. Prior to OS/390 1.3, the number of extents was limited to one; with OS/390 1.3 and later, the number is limited only by the amount of DASD available.

**Attention:** If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference, SA22-7592* for more information about this member.

### **LS\_DATACLAS**

It is very important to remember log stream staging and offload data sets are single extent VSAM linear data sets, and the SHAREOPTIONS **must** be specified as "3,3". If the SHAREOPTIONS are anything other than "3,3" there is a risk the System Logger will be unable to read offloaded data and post CICS with return code 8, reason code 804 (IlgRsnCodeNoBlock). This will cause CICS to abend with a DFHLG0772. Therefore, you should make sure the SMS Data Class specified on the LS\_DATACLAS parameter is set to use SHAREOPTIONS 3,3.

The following messages are additional symptoms of the incorrect SHAREOPTIONS:

DFHLG0772 MVS System Logger codes: X'00000004', X'00000403' or  
DFHLG0772 MVS System Logger codes: X'00000008', X'0000084A'.

With either of the above messages, an associated IEC161I message will be produced.

IEC161I 052(009)-084 error opening a SHAREOPTIONS 1,x data set or  
IEC161I 052(015)-084 error opening a SHAREOPTIONS 2,x data set.

In all cases, the log stream will be considered broken and CICS will terminate.

Following application of OW54863, the System Logger will check the SHAREOPTIONS of the staging and offload data sets at connect time. If the SHAREOPTIONS are not set to (3,3), message IXG267I will be issued. If the System Logger is unable to open the data set, message IXG288I will be issued.

Either an IDCAMS LISTCAT or ISMF display may be used to verify the SHAREOPTIONS are set to 3,3.

## **LOGGERDUPLEX**

LOGGERDUPLEX specifies whether System Logger will continue to perform its own log data duplexing, or conditionally not provide its own duplexing based on an alternative duplexing configuration. An “alternate duplexing configuration” means System-Managed Duplexing is being used for the associated structure.

LOGGERDUPLEX(COND) indicates the System Logger will provide its own duplexing of the log data unless the log stream is being duplexed with System-Managed Duplexing *and* the duplexing is providing the required level of failure independence.

For a more detailed discussion, refer to “Duplexing log data for CF-Structure based log streams” on page 42.

## **STG\_SIZE**

For a CF-Structure log stream, STG\_SIZE defines the size of the staging data set to be allocated if STG\_DUPLEX and DUPLEX\_MODE(YES) are specified. If STG\_DUPLEX(UNCOND) is specified, the data in the CF-Structure log stream will always be duplexed to the staging data set. If STG\_DUPLEX(COND) is specified, the data in the CF-Structure log stream is duplexed to the staging data set only if the CF becomes volatile or becomes failure dependent.

The size of the staging data set (STG\_SIZE) must be at least large enough to hold as much data as the log stream storage in the CF.

The rule of thumb is STG\_SIZE must be large enough to hold the data written during an activity keypoint interval, plus the length of time of the longest running unit of work.

If you are migrating to CICS TS and starting to use System Logger with CICS for the first time, DFHLSCU should be run against the CICS/ESA 4.1 system log and the output used as a *starting point* for STG\_SIZE and the size used for the CF structure definition.

Space within the staging data set is allocated to log blocks in 4096 byte increments, regardless of the buffer size. So, for example, if the average log block size is 2000 bytes, the effective space in the staging data set is reduced by about 50%. Bear this in mind when setting the size of the staging data set.

**Attention:** When staging data sets are used with a CF-Structure log stream, STG\_SIZE must be specified large enough to hold *at least* as much data as the CF-Structure log stream.

The structure storage is divided equally among the active log streams. If the number of log streams connected to the structure will vary, the STG\_SIZE value should be chosen based on the least number of log streams which will normally be connected.

Offload processing is triggered based on the smaller capacity of either the structure log stream storage or the staging data set.

## **STG\_DUPLEX**

STG\_DUPLEX(YES) with DUPLEXMODE(COND) means if the CF becomes volatile, or resides in the same failure domain as the System Logger, the log stream data will be duplexed to the staging data set; otherwise it is duplexed to buffers in the System Logger data space. A CF is in the same failure domain when the CF LPAR and the z/OS LPAR reside in the same physical hardware box, or if the CF is volatile regardless of where it is. Duplexing to the staging data set means the cost of an I/O will be incurred for each write. Therefore we strongly recommend you provide a configuration where the structure containing the log

stream for a CICS region is not in the same failure domain as the system where CICS is executing. To determine if System Logger is using staging data sets for a log stream, use the D LOGGER,LOGSTREAM,LSN= command as shown in Figure 5-10. In the figure, DUPLEXING: LOCAL BUFFERS indicates that the duplex copy of log stream IYOT1.CICS22.DFHLOG resides in System Logger's data space.

```

IXG601I  18.55.09  LOGGER DISPLAY 779
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----          -
IYOT1.CICS22.DFHLOG  LOG_JG2_5M          000001  IN USE
SYSNAME: SYSD
DUPLEXING: LOCAL BUFFERS
  
```

Figure 5-10 Displaying if staging data sets are in use

**OFFLOADRECALL**

OFFLOADRECALL should be set to *NO* for CICS log streams. This option was added via APAR OW48404. This parameter applies in the situation where the System Logger is to perform an offload of data and the target data set has been migrated by DFSMSHsm. Specifying NO indicates to the System Logger a new offload data set is to be allocated rather than attempting a recall. This prevents the System Logger from waiting for the recall and avoids the risk of CICS being unable to write to a log stream which has filled waiting for the offload to proceed.

## DASD-only log stream

Figure 5-11 shows an example of a DASD-only log stream definition. The DASDONLY and MAXBUFSIZE parameters are applicable only for a DASD-only log stream (MAXBUFSIZE is specified at the structure level for CF-Structure log streams).

<pre>//DEFSTR JOB CLASS=A,MSGCLASS=A //POLICY EXEC PGM=IXCMIAPU //SYSPRINT DD SYSOUT=A //SYSIN DD * DATA TYPE(LOGR) REPORT(YES) DEFINE LOGSTREAM NAME(IYOT2.DFHLOG) AUTODELETE(NO) * DASDONLY(YES) DIAG(YES)  EHLQ(NO_EHLQ)  HIGHOFFLOAD(80) *  HLQ(GRAUEL) LOWOFFLOAD(40) *  LS_DATACLAS(LS10MEG) * LS_SIZE(4500) *  MAXBUFSIZE(65532) * MODEL(NO) OFFLOADRECALL(NO) * RETPD(0) * STG_SIZE(9000) *</pre>	<p>Matches name defined in Journal Model</p> <p>Always NO for DFHLOG and DFHSHUNT</p> <p>CF structures will not be used</p> <p>If an 804 (IXGRSNCOENOBLOCK) reason code will be given, a dump of the logger is taken before returning to CICS</p> <p>Allows specification of 33 characters for a high level qualifier - mutually exclusive with HLQ</p> <p>Threshold, expressed as a percent of the logstream space, when offloading is to take place</p> <p>HLQ of the offload data set name</p> <p>Value, expressed as a % of the space used by the logstream, which defines the offload target</p> <p>SMS class used for DASD offload</p> <p>The number of 4K control intervals to be allocated for each offload data set</p> <p>Max buffer size of blocks written to the log</p> <p>Always set to NO for CICS logstreams</p> <p>Retention period, zero for DFHLOG and DFHSHUNT</p> <p>The number of 4K control intervals in the Staging data set</p>
<p>* parms which affect performance</p>	

Figure 5-11 Defining a DASD-only log stream

The following discussion presents the parameters which affect DASD-only log stream performance in a CICS Transaction Server environment.

### **AUTODELETE and RETPD**

AUTODELETE and RETPD can have a disastrous effect on the DFHLOG and DFHSHUNT if specified other than AUTODELETE(NO) and RETPD(0). With AUTODELETE(YES) and RETPD>0, even though CICS will attempt log tail management, all data will be offloaded to the offload data sets and held for the number of days specified for RETPD.

AUTODELETE(YES) lets the System Logger (rather than CICS) decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified, the System Logger will delete the data in the old offload data set. If CICS needs the data for backout, the result will be an 804 return code and CICS will terminate with a DFHLG0772.

For the forward recovery log stream and log of logs, the recommended values are AUTODELETE(NO) and RETPD(xx) where xx is the number of days the data needs to be kept before allowing any deletion. This value is mainly based on the installation business needs. If you are planning to keep this data for a long period, be prepared to manage offload data sets.

## **DIAG**

DIAG(YES) should always be specified for DFHLOG and DFHSHUNT. In a case where the System Logger is unable to locate data requested by CICS, a return code 8 with reason code IXGRSNCODENOBLOCK (804) is presented to CICS. This means backout data is not available and CICS treats it as a fatal error, terminating the region with a DFHLG0772. In many cases, information from the System Logger address space is needed to resolve the problem. When DIAG(YES) is specified in the log stream definition, a dump of the System Logger address is provided (by the System Logger) in addition to the DFHLG0772 dump provided by CICS. There is no overhead associated with this parm unless a dump is requested. This error is normally seen when CICS issues an IXGBRWSE for backout data or during activity keypoint processing when CICS issues the IXGDELET command to trim the tail of the log. It can also be returned during region startup when an IXGCONN (connect) is issued to connect to the log stream.

## **HIGHOFFLOAD**

The HIGHOFFLOAD parameter, in conjunction with the size of the log stream, has a major effect on the amount of storage used by the System Logger. Before data is written to the staging data set, it is written to the System Logger data space.

If the HIGHOFFLOAD value is too high, there can be insufficient space to accommodate data written to the log stream during offload processing. This can lead to a staging data set full condition which causes log writes to be suspended. Also, for DASD-only log streams, space in the staging data set is not freed until the offload completes. Therefore, it is very important to factor in the peak times when calculating the staging data set size and HIGHOFFLOAD values.

We recommend setting HIGHOFFLOAD to 80-85% for all CICS log streams.

## **LOWOFFLOAD**

The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the CICS environment, the LOWOFFLOAD value should be high enough to retain the data required for backout but low enough to keep the number of offloads to a minimum.

LOWOFFLOAD should be set between 40 and 60% for DFHLOG and DFHSHUNT, and to 0 for user journals, forward recovery logs, and auto journals.

In tests setting LOWOFFLOAD closer to HIGHOFFLOAD (for example, 75 with a HIGHOFFLOAD of 80) some improvement was noted in transaction throughput at a cost of an increase in the storage frames and CPU in the System Logger address space. The storage frames used will normally increase as the size of the staging data set increases. Each region should be evaluated based on its workload.

It should also be noted setting the LOWOFFLOAD to a higher value may mask poorly behaving transactions. It may appear the logstream is working properly (for example little or no data is offloaded) when in fact large amounts of storage are being consumed to retain records which would be offloaded with a lower setting. Remember the goal is to contain the data needed for recovery using a minimum of interim storage, with no data being moved to the offload data sets.

## **LS\_SIZE**

LS\_SIZE defines the allocation size for the *offload* data sets. It should be specified large enough to contain several offloads, possibly as much as a day's worth. DFHLOG and DFHSHUNT should only offload a minimal amount of data. For user journals, *all* data is offloaded.

It's very important to specify LS\_SIZE large enough to limit the number of times an additional data set is allocated, to reduce the exposure to delays during the allocation process. The size to be specified is based on the amount of data written during the prime processing window of the day. The amount of data written can be determined using the SMF Type 88 records produced for the log stream by the System Logger.

If you have log streams which offload large volumes of data, find a balance between a size small enough so the system can always find space, while at the same time, minimizing the number of times new offload data sets need to be allocated.

If the extent size of the LS\_DATACLAS (or the value specified in LS\_SIZE) is too small, frequent DASD shifts (allocation of a new offload data set) will occur. Frequent DASD shifts have a negative impact on performance and expose the system to a depletion of DASD space. The number of offload data sets is limited by the DSEXTENT value specified in the LOGR Couple Data Set. The DSEXTENT value defines the number of System Logger directory entries. Each directory entry can point to a maximum of 168 offload data sets. Prior to OS/390 1.3, the number of extents was limited to 1; with OS/390 1.3 and above, the number is limited only by the amount of DASD available.

**Attention:** If LS\_SIZE is not specified in the log stream definition, and a size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

### **LS\_DATACLAS**

It is very important to remember log stream staging and offload data sets are single extent VSAM linear data sets, and the SHAREOPTIONS **must** be specified as "3,3". If the SHAREOPTIONS are anything other than "3,3" there is a risk the System Logger will be unable to read offloaded data and post CICS with return code 8, reason code 804 (IxrRsnCodeNoBlock). This will cause CICS to abend with a DFHLG0772. Therefore, you should ensure the SMS Data Class specified on the LS\_DATACLAS parameter is set to use SHAREOPTIONS 3,3.

The following messages are additional symptoms of the incorrect SHAREOPTIONS:

DFHLG0772 MVS System Logger codes: X'00000004', X'00000403' or  
DFHLG0772 MVS System Logger codes: X'00000008', X'0000084A'.

With either of the above messages, an associated IEC1611 message will be produced.

IEC1611 052(009)-084 error opening a SHAREOPTIONS 1,x data set or  
IEC1611 052(015)-084 error opening a SHAREOPTIONS 2,x data set.

In all cases, the log stream will be considered broken and CICS will terminate.

Following application of OW54863, the System Logger will check the SHAREOPTIONS of the staging and offload data sets. If the SHAREOPTIONS are not set to (3,3), message IXG267I will be issued. If the System Logger is unable to open the data set, message IXG288I will be issued.

Either an IDCAMS LISTCAT or ISMF display may be used to verify the SHAREOPTIONS are set to 3,3.

### **STG\_SIZE**

STG\_SIZE is specified based on the amount of data to be contained in the log stream. The rule of thumb is it must be large enough to hold the data written during a peak activity



keypoint interval, plus the length of time of the longest running unit of work, plus a buffer to allow for higher than normal transaction rates.

If you are migrating to CICS TS and starting to use System Logger with CICS for the first time, DFHLSCU should be run against the CICS/ESA 4.1 system log and the output used as a *starting point* for STG\_SIZE.

**Attention:** If STG\_SIZE is not specified in the log stream definition and a size is not specified in the SMS data class referred to by STG\_DATACLAS, the value is taken from the ALLOCxx parmlib member. The default value in ALLOCxx is 2 tracks, which would result in very frequent offloads for that log stream. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

### **MAXBUFSIZE**

MAXBUFSIZE may be specified for a DASD-only log stream, it defines the largest block which can be written to the log stream. The default value is 65532 and should be used in a CICS environment.

MAXBUFSIZE is returned to CICS at connect time and CICS in turn uses this value as the size of its internal log buffers. For user journals, where the application does not use the wait option, it may be advantageous to specify a smaller size, as the buffer is only written to the log stream when it fills.

### **STG\_DUPLEX**

STG\_DUPLEX(YES) with DUPLEXMODE(COND) are *not applicable* to DASD-only log streams.

### **OFFLOADRECALL**

OFFLOADRECALL should be set to NO for CICS log streams. This option was added via z/OS APAR OW48404. Specifying NO indicates to System Logger if a required offload data set has been migrated by DFSMSHsm, then a new offload data set is to be allocated. This prevents System Logger from waiting for the recall, which could result in the staging data set filling before the offload completes.

**Attention:** With the exception of DFHLOG and DFHSHUNT, DASD-only log streams can be shared within the z/OS image. User journals and forward recovery logs fall into this category. Please note, this could have an impact on IOSQ time to the related staging data sets.

Another important point, if you have a DASD-only user journal shared between multiple CICS regions in the same image, then you must ensure all the CICS regions sharing the journal always run on the same z/OS image. If one CICS is moved to another image, only the CICS regions on the first z/OS image to connect to the log stream will be successful.

## MVS Model log streams - DFHLOG

Figure 5-12 illustrates definitions for a CF-Structure log stream model for DFHLOG. If the log stream is to be a DASD-only log stream, the CF-specific parms would be replaced by those which pertain to DASD-only.

<pre>//DEFMODEL JOB CLASS=A,MSGCLASS=A //MODDEF EXEC PGM=IXCMIAPU //SYSPRINT DD SYSOUT=* //SYSIN DD * DATA TYPE(LOGR) REPORT(YES) DEFINE LOGSTREAM NAME(sysname.DFHLOG.MODEL) STRUCTNAME(LOG_DFHLOG_001) DIAG(YES)  HIGHOFFLOAD(80)  LOWOFFLOAD(40)  LS_DATACLAS(LS10MEG) OFFLOADRECALL(NO) STG_DUPLEX(YES) DUPLEXMODE=(COND)  STG_SIZE(9000) MODEL(YES) LS_SIZE(100000)</pre>	<p>sysname - name of the z/OS image Structure which will contain the data If an 804 (IXGRSNCODENOBLOCK) reason code will be given, a dump of the logger is taken before returning to CICS Threshold, expressed as a % of logstream space, when offloading is to take place Value, expressed as a % of the space used by the logstream, of the offload target SMS dataclass used for offload data sets Always set to NO for CICS logstreams Log writes to be duplexed to the CF and Staging data sets if CF becomes volatile Number of 4K CIs in the Staging data set A model only, connection is not permitted The number of 4K blocks for offload data sets allocation MSG IXG256I is issued if the value is less than 64K</p>
--	---

Figure 5-12 Defining a MODEL log stream for DFHLOG

The MVS model associates a log stream with the desired characteristics. Model usage makes it easier to dynamically define CICS log streams. The disadvantage is all log streams using the model have the same characteristics.

## MVS Model log streams - User journals

Figure 5-13 on page 169 illustrates definitions for an MVS model for a user journal log stream.

Models for user journals are named based on the first *two* qualifiers in the log stream name. For example, the model in Figure 5-13 would be used for user journals such as IYOT1.IYO1.DFHJ03.

<pre>//DEFMODEL JOB CLASS=A,MSGCLASS=A //MODDEF EXEC PGM=IXCMIAPU //SYSPRINT DD SYSOUT=* //SYSIN DD * <b>DATA TYPE(LOGR) REPORT(YES)</b> <b>DEFINE LOGSTREAM NAME(IYOY1.IYO1.MODEL)</b> <b>DASDONLY(YES)</b> <b>AUTODELETE(Y)</b>  <b>RETPD(15)</b> <b>DIAG(YES)</b>  <b>HIGHOFFLOAD(85)</b>  <b>LOWOFFLOAD(00)</b>  <b>LS_DATACLAS(LS10MEG)</b> <b>OFFLOADRECALL(NO)</b> <b>MODEL(YES)</b> <b>STG_SIZE(5000)</b></pre>	<p><b>Logger will delete the data when the retention period (RETPD) has elapsed.</b></p> <p><b>Retain the data for 15 days</b></p> <p>If an 804 (IXGRSNCODENOBLOCK) reason code will be given, a dump of the logger is taken before returning to CICS</p> <p><b>Threshold, expressed as a % of logstream space, when offloading is to take place</b></p> <p><b>Value, expressed as a % of the space used by the logstream, of the offload target</b></p> <p><b>SMS dataclass used for offload data sets</b></p> <p><b>Always set to NO for CICS logstreams</b></p> <p><b>A model only, connection is not permitted</b></p> <p><b>Number of 4K CIs in the staging data set</b></p>
---	---

Figure 5-13 Defining a MODEL log stream for a user journal

The example shows a DASD-only log stream; if a Coupling Facility log stream is required, replace the DASD-only parameters with those for a CF-Structure log stream.

## 5.3 Operational considerations

Following are some operational considerations for the CICS environment:

### 5.3.1 RACF authorizations

For the CICS user, authorizations to access System Logger resources fall into two categories: definition and use of System Logger resources.

**Attention:** Refer to “Required access levels” on page 21 for authorization considerations of System Logger address space.

When IXCMIAPU is used to predefine the CICS log streams (including the MVS log stream models) the user requires the following authorizations:

- ▶ ALTER access to RESOURCE(MVSADMIN.LOGR) CLASS(FACILITY)
 

ALTER access to this profile is required to delete and define log structures in the System Logger policy. For users who only want to read the definitions, READ access is sufficient. For example:

```
PERMIT MVSADMIN.LOGR CLASS(FACILITY) ACCESS(ALTER) ID(your_userid)
```

- ▶ UPDATE access to RESOURCE(MVSADMIN.XCF.CFRM) CLASS(FACILITY)

UPDATE access to this profile is required to define structures in the CFRM policy. For example:

```
PERMIT MVSADMIN.XCF.CFRM CLASS(FACILITY) ACCESS(ALTER) ID(your_userid)
```

- ▶ ALTER access to RESOURCE(log\_stream\_name) CLASS(LOGSTRM)

ALTER access to this profile is required to define, update, and delete both DASD-only and Coupling Facility log streams.

For example:

```
RDEFINE LOGSTRM log_stream_profile UACC(NONE) [NOTIFY]
PERMIT log_stream_profile CLASS(LOGSTRM) ACCESS(ALTER) ID(your_userid)
```

- ▶ UPDATE access to RESOURCE(IXLSTR.structure\_name) CLASS(FACILITY)

UPDATE access to this profile is required to specify a structure name on the log stream definition.

```
RDEFINE FACILITY IXLSTR.structure_name_a UACC(NONE) [NOTIFY]
PERMIT IXLSTR.structure_name_a CLASS(FACILITY) ACCESS(UPDATE) ID(your_userid)
```

- ▶ ALTER access to RESOURCE(IXLSTR.structure\_name) CLASS(FACILITY)

ALTER access to this profile is required to define and delete CF structures in the System Logger policy.

For example:

```
RDEFINE FACILITY IXLSTR.structure_name_a UACC(NONE) [NOTIFY]
PERMIT IXLSTR.structure_name_a CLASS(FACILITY) ACCESS(ALTER) ID(your_userid)
```

For CICS regions which dynamically create (autoinstall) log streams using the IXGINVNT service, the CICS region user ID must have the following authorization:

- ▶ ALTER access to RESOURCE(log\_stream\_name) CLASS(LOGSTRM)

ALTER access to this profile is required in order for CICS to be able to define new log streams in the System Logger policy. A profile is required for each log stream. It may be more efficient to use a generic profile by region such as *region\_userid.applid.\** where *region\_userid.applid* are the first two qualifiers of the log stream names.

- ▶ UPDATE access to RESOURCE(IXLSTR.structure\_name) CLASS(FACILITY)

UPDATE access to this profile is required to allow CICS to specify a structure name when it dynamically creates a log stream.

For example, consider the log streams are to be named *region\_userid.applid.DFHLOG* and *region\_userid.applid.DFHSHUNT*, the following permissions would allow the CICS region to define DFHLOG and DFHSHUNT as Coupling Facility log streams. The access is defined as follows:

```
PERMIT region_userid.applid.* CLASS(LOGSTRM) ACCESS(ALTER) ID(region_userid)
PERMIT IXLSTR.structurename CLASS(FACILITY) ACCESS(UPDATE) ID(region_userid)
```

If the log streams to be used by the CICS region have been predefined to the System Logger, CICS is only required to have UPDATE authority to the log stream profiles in order to be able to use them. For example:

```
PERMIT region_userid.applid* CLASS(LOGSTRM) ACCESS(UPDATE) ID(region_userid)
```

Refer to the following documentation for additional information:

- ▶ The section entitled “Define Authorization to System Logger resources” in *z/OS MVS Setting Up a Sysplex, SA22-7625*

- ▶ The section entitled “Authorizing access to MVS log streams in the *CICS Transaction Server for z/OS Installation Guide*, GC34-5985
- ▶ The section entitled “Security for Journals and log streams” in the *CICS Transaction Server for z/OS RACF Security Guide*, SC34-6011

### 5.3.2 Data set lifetime

From an operational point of view it's important to understand the life and naming conventions of the various log stream data sets associated with the CICS log streams.

#### **Staging Data Sets**

The High Level Qualifier (HLQ) is specified on the log stream definition. System Logger Staging data sets used with a DASD-only log stream are of the form:

HLQ.LSN.sysplex\_name

The *sysplex\_name* is specified as SYSPLEX() in the COUPLExx member in SYS1.PARMLIB. The *sysplex* name is used to make the data set name unique within the sysplex, ensuring only one system at a time can use the log stream. For example:

```
PRODSYSA.IYOT1.DFHLOG.PLEXB
- PRODSYSA is the High level Qualifier
- IYOT1.DFHLOG is the log stream name
- PLEXB is the sysplex name
```

Staging data sets used with a Coupling Facility log stream are of the form:

HLQ.LSN.system\_name

The *system\_name* is specified in the IEASYSxx, IEASYMxx, or Loadxx members of SYS1.PARMLIB. The system name identifies the z/OS image within the sysplex.

```
PRODSYSA.IYOT1.DFHLOG.MV55
- PRODSYSA is the High Level Qualifier
- IYOT1.DFHLOG is the log stream name
- MV55 is the z/OS image name
```

Staging data sets are allocated when the first exploiter connects to the log stream, and exist only for the period of time it is connected to the log stream. When the last connector disconnects, the data is offloaded and the staging data set deleted.

#### **Offload data sets**

Offload data sets, sometimes referred to as log data sets, are named as follows:

```
HLQ.LSN.sequence#
PRODSYSA.IYOT1.DFHLOG.A0000003
- PRODSYSA is the High Level Qualifier
- IYOT1.DFHLOG is the log stream name
- A0000003 is the sequence number of the offload data set assigned by the System
Logger
```

The sequence number begins with A0000000 when the first offload data set is allocated (this happens as soon as you add the log stream definition to the System Logger policy). The number will be incremented each time a new offload data set is allocated to the log stream. If/when the log stream is deleted and redefined, the number will be reset to A0000000.

CICS knows nothing about the offload data sets and an INITIAL start of CICS does not alter/reset the sequence number.

Figure 5-14 is an example showing an ISPF display of the offload data sets for IYOT1.DFHLOG. Note this is a DASD-only log stream (the staging data set lowest qualifier is the sysplex name) and CICS is currently connected (staging data sets only exist for the duration of time CICS is connected to the log stream).

```

DSLISL - Data Sets Matching GRAUEL.IYOT1.DFHLOG

          Command - Enter "/" to select action          Message          Volume
-----
OFFLOAD data set  GRAUEL.IYOT1.DFHLOG.A0000003                *VSAM*
                  GRAUEL.IYOT1.DFHLOG.A0000003.DATA      PBDA02
STAGING data set  GRAUEL.IYOT1.DFHLOG.PLEXB                *VSAM*
                  GRAUEL.IYOT1.DFHLOG.PLEXB.DATA         PBDA06

LSN = LogStream Name (IYOT1.DFHLOG)
HLQ = High level qualifier - specified on logstream definition (GRAUEL)
A0000003 is the offload data set sequence number
PLEXB is the sysplex name indicating this is a staging data set for a DASDONLY logstream

```

Figure 5-14 Staging and offload data sets for IYOT1.DFHLOG log stream

The RETPD parameter on the log stream definition, allows you to specify a retention period for log stream data. With RETPD, you specify the number of days the data in the log stream is to be kept, even if the data has been marked for deletion using an IXGDELET call.

For example, if you specify RETPD(7) in the System Logger policy for a log stream, the retention period for *data* in that log stream is 7 days from the time the data is written to the log stream. System Logger processes the retention period on a offload data set basis. Once the retention period for all the data in the offload data set has expired, the data set is eligible for deletion. The data might not be physically deleted as soon as it hits the retention period. The point at which the data is physically deleted depends on when the data set fills and the specification of the AUTODELETE parameter. To understand when the data sets will be physically deleted, refer to 2.7.2, “When is my log data or offload data set physically deleted?” on page 65.

**Note:** The System Logger retention period applies to the age of the log data, not the data set.

AUTODELETE(NO) means the log data can be physically deleted only after the data has been marked for deletion via an IXGDELET call *and* after any retention period specified for the log stream has expired. AUTODELETE(NO) is the default.

AUTODELETE(YES) means log data can be physically deleted *either* when the data is marked for deletion (by IXGDELET) *or* when a retention period specified for the log stream data expires. AUTODELETE(YES) is designed to speed physical deletion of log data.

Specifying anything other than AUTODELETE(NO) and RETPD(0) for DFHLOG and DFHSUNT can have a disastrous effect on CICS, affecting both performance and availability. With RETPD>0, even though CICS attempts log tail management, all data will be offloaded to the offload data sets and held for the number of days specified for RETPD.

AUTODELETE(YES) lets System Logger (rather than CICS) decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified without a retention period, System Logger will delete the data on the old offload data set. If CICS needs the data for backout, the result will be an 804 return code from System Logger and CICS will terminate with a DFHLG0772.

As for the forward recovery or user journal log stream, it is usually suggested to use AUTODELETE(NO) and to specify a value as RETPD. Specifying a value as RETPD ensures the records are going to be kept for this period. AUTODELETE(NO) should be used because both forward recovery and user journal records are processed by utilities and afterwards deleted by the utilities themselves.

Figure 5-15 contains a summary of when the offload data sets are deleted, based on the AUTODELETE and RETPD specifications.

AUTODELETE(NO) RETPD(0)	Log data sets are eligible for deletion when <u>all data has been deleted</u> via an IXGDELET call. This combination must always be used for DFHLOG and DFHSHUNT.
AUTODELETE(NO) RETPD(xx)	All data is offloaded from interim storage to the offload data sets. Offload data sets are eligible for deletion when <u>all data has been deleted</u> via an IXGDELET call <u>and the retention period has expired</u> . This combination should <u>not</u> be used for DFHLOG and DFHSHUNT.
AUTODELETE(YES) RETPD(xx)	All data is offloaded from interim storage. Offload data sets are eligible for deletion when <u>either all the data has been deleted or the retention period has expired</u> . This combination should <u>not</u> be used for DFHLOG or DFHSHUNT.
AUTODELETE(YES) RETPD(0)	Offload data sets are deleted <u>when the next offload data set is allocated</u> . This combination should <u>not</u> be used for DFHLOG and DFHSHUNT as it is possible for recovery data to be deleted, making backout impossible.

Figure 5-15 Offload data set deletion

### 5.3.3 HSM considerations

Many installations use DFSMSHsm to manage the offload data sets. However, caution should be used when allowing DFSMSHsm to manage any System Logger data sets. Staging data sets and the offload data sets for DFHLOG and DFHSHUNT should *not* be under the control of DFSMSHsm.

If SETSYS INTERVALMIGRATION has been specified in the ARCCMDxx member, DFSMSHsm will, on a regular interval (default is 60 minutes), examine the managed volumes to see if the allocated space is over the specified high threshold. If the high threshold has been reached, it will begin the migration of data sets to reach the low threshold value.

When DFSMSHsm is about to migrate (or restore) a data set, it holds the serialization lock (ENQ) for the file. If System Logger attempts to allocate or connect to the file at this time, Media Manager (a component of DFSMS) will return an error. The error as presented does not indicate a temporary condition, so it's treated as though the file is inaccessible and a gap-type error or an offload failure is returned to CICS depending on which operation is in progress.

Examples would be return code 8, reason code 84A, or return code 4 with reason code 403.

*CICS treats this as a log integrity error and terminates with message DFHLG0772.*

This is another manifestation of the SHAREOPTIONS problems as defined earlier in "LS\_SIZE" on page 160 for Coupling Facility log streams or "LS\_SIZE" on page 165 for DASD-only log streams.

### 5.3.4 Disaster recovery

Disaster recovery brings in a new set of considerations for the CICS environment. Log streams can *not* be simply transported using a volume dump of the volume containing the staging or offload data sets.

There are options to enable disaster recovery of CICS systems using log streams. The first is to simply carry the CICS libraries to the remote site, define (or delete and redefine) the log streams and issue an INITIAL start of CICS. However any transactions in-flight at the point of failure would not be backed out, leaving an exposure for data issues.

The other option is to use a disaster recovery product such as Geographically Dispersed Parallel Sysplex™ with Peer to Peer Remote Copy (GDPS/PPRC) or Extended Remote Copy (GDPS/XRC) to propagate the DASD data to the remote site. Using either of these products allows the mirroring of staging data sets, the LOGR CDS, the z/OS catalogs, all CICS data sets, and all the System Logger data sets to provide a consistent state at the disaster recovery site. Only when the secondary site has the data in a consistent state can an emergency restart of the CICS regions be performed.

It is important to note Coupling Facility log stream *must* be duplexed to staging data sets in order for this process to work.

#### **Peer to Peer Remote Copy - PPRC**

PPRC is a synchronous data mirroring technology; data is committed to both primary and secondary subsystems before the write is confirmed to be complete. As a consequence, this technology is sensitive to performance impact as the distance between primary and secondary subsystems increases. The maximum distance in a GDPS® environment is around 40 km. This limit is imposed by the Sysplex timer configuration - if you are only performing remote mirroring without GDPS, the distance between primary and secondary disk subsystems could be up to 103 km (IBM Enterprise Storage System).

With a synchronous mirroring technology, secondary data consistency does not come automatically, it needs to be managed. This is one of the main reasons for the existence of GDPS. GDPS/PPRC is a Continuous Availability solution, making major improvements to application availability by automating error detection and error response.

#### **Extended Remote Copy - XRC**

Due to risks, mostly related to environmental hazards such as earthquakes, blizzards, hurricanes, or flooding, some enterprises need disaster protection with a failover site outside their primary site region. In this case, ESCON® or even FICON® technology might no longer be feasible and telecommunication facilities must be used so the two IT facilities can be connected over large distances.

Propagation delays resulting from long distances mean in many cases synchronous data mirroring would have a large performance impact. This is one of the main reasons for the existence of eXtended Remote Copy (XRC), a software-based data mirroring solution which operates asynchronously.

It is important to remember since the copy technology is asynchronous, some data loss is to be expected in a failover. On the other hand, the design of XRC ensures the data at the remote site is time consistent, whereas this is not guaranteed with PPRC.



### 5.3.5 Log stream sizing

Log stream data exists in temporary or interim storage, and in permanent or hardened storage. This section discusses sizing of the interim storage in order to achieve the best log stream and overall system performance. The discussion will include:

- ▶ Sizing log streams when migrating from CICS 4.1
- ▶ Using the DFHLSCU utility
- ▶ Sizing DASD-only log streams
- ▶ Sizing CF-Structure log streams

For information about sizing the permanent storage, refer to the discussion about “LS\_SIZE” on page 160 for CF-Structure log streams and “LS\_SIZE” on page 165 for DASD-only.

There are many variables which must be taken into account when sizing a log stream, making it difficult to get it perfect the first time. Once an initial size is determined, the log stream performance must be monitored using the SMF Type 88 records produced by the System Logger, and adjustments made as required.

Understanding the volume of data and the rate at which it is written is critical when sizing the log stream interim storage.

At this point, a review of the basic System Logger concepts found in Chapter 1, “Introduction to System Logger” on page 1 is in order.

#### Interim storage and duplexing

For a complete description of the interim storage and the different way of duplexing data, refer to 2.5, “Log streams” on page 22. Following are the considerations which affect the CICS logging environment.

To optimize use of the data space, and reduce the potential impact on z/OS storage, residency time of the data should be kept to a minimum. For the CICS system log (DFHLOG and DFHSHUNT), the data should be retained in interim storage until the associated UOW commits or backs out. For non-system logs, all data is offloaded to the offload data sets so the interim storage (CF structure or staging data set) should be sized to provide minimal residency time.

Offload data sets contain data which has been offloaded from interim storage. Data is offloaded from interim storage as part of the offload process, usually initiated when the HIGHOFFLOAD threshold is reached. Offload data sets may be subsequently archived using a product like DFSMSHsm.

#### Sizing interim storage

**Important:** Remember, sizing a logstream is an iterative process. Unfortunately, it's difficult to get it perfect the first time.

Once a starting size has been identified, the SMF Type 88 records should be used to tune the log stream based on the activity during the key processing intervals.

When calculating the size of interim storage for a log stream, the following log stream characteristics are important:

- ▶ The number of IXGWRITE requests issued during a specified time interval.
  - CICS requests the System Logger to record information in a log stream using an IXGWRITE call. Each buffer passed may contain multiple CICS log records.

- The number of IXGWRITE requests is found in the CICS log stream statistics (LGSWRITES), and in the SMF Type 88 records for the log stream (SMF88LWB or IXGWRITES).
- ▶ Number of bytes written in the interval.
  - The number of bytes written is found in the CICS log stream statistics (LGSBYTES) and in the SMF Type 88 records (SMF88SWB) written for the log stream.
- ▶ Number of bytes written per IXGWRITE, or average buffer size.
  - Bytes written divided by the number of write requests (LGSBYTES/LGSWRITES).
  - Reported as the average buffere size in the SMF 88 reports produced with IXGRPT1.
- ▶ HIGHOFFLOAD percentage. This determines at which point the offload will start and also determines how much space is left between when the offload starts and when the log stream fills.
- ▶ The number of log stream deletes for DFHLOG and DFHSHUNT.
  - This is a reflection of the number of times CICS performed log tail management. Log tail management is the process where CICS will issue an IXGDELET call to indicate records which are no longer required for recovery and may be deleted. Log tail management reduces the size required for interim storage without having to offload the data.
  - The number of log stream deletes is found in the CICS log stream statistics (LGSDLETES) and the SMF 88 data (SMF88SII and SMF88SAI).
- ▶ Number of offloads during the interval.
  - Provided in the SMF Type 88 records (SMF88EO).

Each CICS system log stream will require enough storage to hold the *data written in an Activity KeyPoint (AKP) interval + the duration of the longest UOW*. Structures should be large enough to hold the sum of the data written for all connected log streams in the interval.

When migrating to CICS Transaction Server from an older release of CICS, the Log Stream sizing migration utility (DFHLSCU) should be run against the system journal (DFHJ01) and each user journal from the older release. DFHLSCU is discussed further in “DFHLSCU - Log Stream Sizing utility” on page 186.

If pre-CICS Transaction Server journals are available, skip to “DFHLSCU - Log Stream Sizing utility” on page 186.

### ***Coupling Facility Control Code considerations***

With each release of the Coupling Facility Control Code (CFCC), the minimum structure size has increased. In CF Level 12, the minimum size is now 4608K. If a structure smaller than this size is being used for CICS system log streams (DFHLOG or DFHSHUNT), CICS may fail to initialize following an upgrade to CF Level 12.

Example 5-1 shows the error messages received when DFHLOG was allocated to a 2 MB structure, when attempting to initialize CICS at CF Level 12.

#### ***Example 5-1 Example of inability to connect to a small structure***

---

```

16.51.33 JOB12509 +DFHLG0103I IYOT1 System log (DFHLOG) initialization has started.
16.51.34 JOB12509 +DFHLG0786 IYOT1 173
The MVS logger has returned an error during operation IXGCONN CONNECT
for log stream IYOT1.CICS22.DFHLOG. The MVS logger failed to find a
suitable coupling facility for the log stream structure. MVS logger
codes: X'00000008' X'00000853'.
```

```

+DFHLG0103I IYOT1 System log (DFHLOG) initialization has started.
IEF196I IXG206I CONNECT FAILED FOR LOGSTREAM IYOT1.CICS22.DFHLOG
IEF196I IN STRUCTURE LOG_JG_2M. NO SUITABLE COUPLING FACILITY FOUND.
IXG206I CONNECT FAILED FOR LOGSTREAM IYOT1.CICS22.DFHLOG 163
IN STRUCTURE LOG_JG_2M. NO SUITABLE COUPLING FACILITY FOUND.
IEF196I IXG207I CF NAME: CF1 REASON CODE: 00000007 CF MINIMUM SIZE:
IEF196I 4608K BYTES.
IXG207I CFNAME: CF1 REASON CODE: 00000007 CFMINIMUMSIZE: 4608K BYTES.
IEF196I IXG207I CF NAME: CF2 REASON CODE: 00000007 CF MINIMUM SIZE:
IEF196I 4608K BYTES.
IXG207I CF NAME: CF2 REASON CODE: 00000007 CF MINIMUM SIZE: 4608K
BYTES.
IXL013I IXLCONN REQUEST FOR STRUCTURE LOG_JG_2M FAILED. 161
JOBNAME: IXGLOGR ASID: 0013 CONNECTOR NAME: IXGLOGR_SYSD
IXLCONN RETURN CODE: 0000000C, REASON CODE: 02010C08
CONADIAGO: 00000002
CONADIAG1: 00000008
CONADIAG2: 00000C08
+DFHLG0786 IYOT1 173
The MVS logger has returned an error during operation IXGCONN CONNECT
or log stream IYOT1.CICS22.DFHLOG. The MVS logger failed to find a
suitable coupling facility for the log stream structure. MVS logger
codes: X'00000008' X'00000853'.
+DFHLG0731 IYOT1 A failure has occurred while opening the system log
(DFHLOG). CICS will be terminated.

```

---

### ***When DFHLSCU is not used for sizing***

If there are no pre-CICS Transaction Server journals available, the use of DFHLSCU is not appropriate. In those cases, the methodology used to size the DFHLOG and DFHSHUNT log streams would follow the process listed in the following example. Using this method may result in oversizing the log stream which should then be reduced based on the activity reported in the SMF Type 88 records.

The non-system logs (user journals, forward recovery logs, and auto journals) are considered to be funnel-type log streams (all data is offloaded). The size of the interim storage should be much smaller than the space allocated to the log data set in CICS V4.1. Oversizing non-system logs increases the working set of the System Logger and may result in overall system paging problems.

### ***DFHLOG and DFHSHUNT example***

Consider a hypothetical transaction which causes 42 log records to be written each time it is executed.

To be able to arrive at a ballpark value for the amount of interim storage required for the log stream, collect data relating to the number of transactions, transaction duration, and the journal activity in the CICS V4.1 region. The CICS statistics interval should be set to match the SMF interval. In the sample case, the SMF interval and CICS statistics interval were set to five minutes for a CICS V4.1 region. The CICS statistics interval is set using the CEMT command while the SMF interval is set in SYS1.PARMLIB.

When using the z/OS Workload Manager (WLM) in goal mode, it is possible to define a report class for CICS transactions. The report class is used to show the number of transactions ended in the SMF interval, the transaction rate (number of transactions ending per second), and the response time.

The same information can be gathered using a Report Performance Group, if WLM is being used in compatibility mode. For guidance with defining a compatibility mode Report Performance Group, refer to the section entitled “Defining Service Classes and Performance Goals” in *z/SO MVS Planning: Workload Management, SA22-7602*.

Figure 5-16 is an example of the information provided when a report class is used. In this case all JOR\* transactions have been grouped into a report class called RJORIY1 to identify the transactions as having run in the IYOT1 region.

The information presented is for an SMF interval, which has been set to five minutes. The number of transactions which ended in the interval was 16479, the transaction rate was 54.93 per second, and the average transaction duration (response time) was 163 seconds.

REPORT BY: POLICY=WLMPOL		REPORT CLASS=RJORIY1	
DESCRIPTION =Report class -JOR trans IYOT1			
TRANSACTIONS	TRANS.-TIME	HHH.MM.SS.TTT	
AVG	0.00	ACTUAL	163
MPL	0.00	EXECUTION	0
ENDED	16479	QUEUED	0
END/S	54.93	R/S AFFINITY	0
#SWAPS	0	INELIGIBLE	0
EXCTD	0	CONVERSION	0
AVG ENC	0.00	STD DEV	0
REM ENC	0.00		
MS ENC	0.00		

Figure 5-16 RMF Report Class for the JOR\* Transactions in a R410 region

Figure 5-17 provides the interval statistics for journals in a CICS V4.1 region during the same five minute window used in Figure 5-16. The activity for Journal 1 indicates 1039743 records were appended to the buffer, with 196349 blocks written. The average blocksize size was 560 bytes.

Dividing the number of records by the number of transactions (1039743/16479) gives an average of 63 buffer adds per transaction. To be totally accurate, there were 199 CSKP tasks which executed in the interval, but the average would still round to 63.

It's important to understand the number and size of the records in CICS Transaction Server are different than those written in CICS V4.1. So, although these numbers provide some insight, the result will normally not be as accurate as using the DFHLSCU utility.

JOURNALS												
Journal ID	Journal Type	Records Written	Blocks Written	Buffer Full	Ave. O/P Blk size	Last Vol Written	Tapes Opened	Tapes Left	Waits on Archive	Archives Submit.	Datasets Opened	
1	DISK2	1039743	196349	0	560		0	0	0	0	0	
3	DISK2	0	0	0	0		0	0	0	0	0	
<b>*TOTALS*</b>		<b>1039743</b>	<b>196349</b>	<b>0</b>			<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	

Figure 5-17 Sample Journals statistics for a CICS V4.1 region

```

INITSIZE = 4068K + (LOGSNUM * A * B / 1024)
where   A = 2000 + (no. entries + 5)
and     B = (AVGBUFSIZE * 1.1289) + 195

```

**Note:** The original formula used a value of 310 for fixed costs. As function has been added to the CFCC code, the value has increased. Testing with CF Level 12 has shown the minimum System Logger structure size is 4068K. For the purposes of this book, the fixed costs value is being changed to 4068K. Subsequent tests show the resulting structure is slightly larger than required, so the log stream activity should be monitored and the size adjusted to meet the log stream requirements.

Using the formula for INITSIZE, we can estimate the size required using the CICS V4.1 information, based on the following information:

```

Transaction rate      54.93 per second.
Transaction duration  .163 seconds.
63 journal buffer appends per transaction.
654.5 (196349 writes /300 seconds in the interval) journal writes (IXGWRITEs) per sec.
Average buffer size is 560 bytes.
AKPFREQ is set to 4000.

```

When a resource manager calls the CICS logger domain and the record is added to the CICS log buffer -- it is considered a buffer append. AKPFREQ defines the number of buffer appends before an activity keypoint is initiated.

```

akpintvl = AKPFREQ / ((number of transaction in interval * buffer appends per trans)
= (4000 buffer appends) / (54.93 tran/sec * 63 buffer appends/trans)
= 4000/3460.6 = 1.16 seconds or 51.8 times per minute

```

```

no. entries = ((1.16 + 0.163 sec) * (654.5 writes/sec))/0.9
= ((1.323)*(980))/0.9
= ((865.90))/0.9
= 962.1 round to 963

```

```

B = (AVGBUFSIZE * 1.1289) + 195
= (560 * 1.1289)+ 195
= (632.18) + 195
= 827.18

```

```

A= 2000 + (no. entries +5)
= 2000 + (963 +5)
= 2968

```

```

In this sample there will be one log stream in the structure
INITSIZE = 4068K + ((logsnum * A * B)/1024)
= 4068K + ((1 * 2968 * 827.18)/1024)
= 4068K + (2455070.24)/1024
= 4068K + 2397.52K = 6465.52K

```

Remember, the size required is workload dependent. Any deviations in the workload might alter the amount of interim storage required.

There seems to be a additional variable with CF Level 12 based on the maximum structure size (SIZE) specified in the structure definition. Testing using a 5 MB structure, based on the above calculations, with log defer (LGDFINT) set to 5 milliseconds and 0, did not produce the transaction throughput expected.

To eliminate contention on the logstream the structure was expanded to 10M. The results are presented as samples A and B in Example 5-2.

Example 5-2 Sample IXGRPT1 report along with data reported from RMF and DFH0STAT

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES	---# WRITES COMPLETED---			AVERAGE			
		BY USERS	TO INTERIM	TO DASD		TYPE1	TYPE2	TYPE3		BUFFER		
		IXGWITES	STORAGE		INVOKED				SIZE			
		BYT DELETED	# DELETES	BYT DELETED	# DELETES	-----EVENT-----						
		INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-
W/O DASD	WRITE	W/DASD	WRITE	LOAD SHFT	FULL	FULL	THLD	FULL	BLD			
<hr/>												
<b>(A) AKPFREQ 1000 LGDFINT 5 10M structure HIGHOFFLOAD 80 LOWOFFLOAD 50</b>												
12/12/02 0:45:00 AM (SMF INTERVAL TIMESTAMP 'B8A9F9222BB00000'X)												
IYOT1.CICS22.DFHLOG	LOG_JG2_10M	4560403	5881856		0	7417	7371	46	0	614		
		4418914	7190		0	0	6	0	0	0	0	0
*****												
<b>(B) AKPFREQ 1000 LGDFINT 00 10M structure HIGHOFFLOAD 80 LOWOFFLOAD 50</b>												
12/12/02 0:55:00 AM (SMF INTERVAL TIMESTAMP 'B8A9FB5E60100000'X)												
IYOT1.CICS22.DFHLOG	LOG_JG2_10M	320447741	531816704		0	1039948	1025325	14622	0	308		
		320157518	1039384		0	0	592	0	0	0	0	0
<hr/>												

### DASD-only

The following formula calculates the value to be specified on the STG\_SIZE parameter of the log stream definition; that is, the size is expressed as a number of 4K blocks:

Staging DS size [No. of 4K blocks] = (AKP duration) \* No. of log writes per second for system log

where:

AKP duration = (CICS TS 390 AKPFREQ) / (No. of buffer puts per second)

The values for the number of log writes per second and buffer puts per second can be taken from your CICS/ESA 4.1 statistics.

### Sizing non-system log streams

There are slight differences between a User journal, a forward recovery log, and an auto journal. Data on a User journal is written directly by an application, for example when the application issues an EXEC CICS WRITE JOURNALNAME command. A forward recovery log contains "after" images used to recover the file updates which are then applied to an image copy of a VSAM file. Forward recovery logs are used by products such as CICS/VR. An auto journal may contain a combination of records based on the file definition parms JNLADD, JNLREAD, JNLSYNCREAD, JNLSYNCWRITE, JNLUPDATE, and JOURNAL.

There are also subtle differences in the amount of data written to the log/journal. In all cases, each block written will contain a x'28' (decimal 40) byte header. For User journals, CICS appends 68 bytes of information to each record written. For forward recovery and auto journal log streams, 84 bytes of CICS information plus the length of the record key is appended to each log record. Refer to *CICS TS for z/OS Customization Guide*, SC34-5989 for information regarding the record format.

Another difference is the point in time the data is actually written to the log stream. For a User and auto journal, the write happens when the buffer fills. For a forward recovery log, the data is written at sync point.

In the following descriptions, the goal is to define the log stream so residency time is about one minute. To reduce the overall impact to the z/OS image, the residency time should be kept short, therefore reducing the working set for the System Logger.

Several pieces of information are required:

- ▶ MAXBUFSIZE for the log stream.
- ▶ Average log record size and key length.
- ▶ Number of records written in a given interval.
- ▶ The CICS log stream type (USER, auto journal, forward recovery).
- ▶ Physical log stream type (Coupling Facility log stream or DASD-only).

For a DASD-only log stream, data is written in 4K (4096 bytes) control intervals (CI). This means the size of the actual data placed into interim storage is rounded to a 4096 byte boundary. Divide MAXBUFSIZE by 4096 to determine the number of CIs required to hold a full buffer. The number of CIs also represents the number of pages required in the System Logger data space (for both DASD-only and CF-Structure log streams). When MAXBUFSIZE is set at 65532, it will hold up to 16 CIs of data.

For CF-Structure log streams, the data placed into interim storage is rounded to the element boundary (256 or 512 bytes). For additional information refer to "IXGRPT1 observations and possible actions" on page 195.

The suggested process for log stream sizing:

- ▶ Determine MAXBUFSIZE
  - For DASD-only, this is specified on the log stream definition (65532 is recommended to optimize the I/O).
  - For a CF-Structure log stream, it is specified on the structure definition in the System Logger policy (for CICS log streams, the recommended value is 64000).
  - CICS adds a 40 byte header to each block written - so the usable size is (MAXBUFSIZE - 40 bytes).
- ▶ Determine the record size
  - For a user journal -- data record + 68 bytes of CICS description.
  - For an auto journal -- data record + keylength + 84 bytes for CICS descriptor.
  - For a forward recovery log - data record + keylength + 84 bytes for CICS descriptor.
- ▶ Identify the amount of data written within an interval
  - (transaction rate per second)\*(records per transaction)\*(record size).
- ▶ Establish the number of data CIs created per second
  - # of data CIs = the amount of data written in one second, divide by 4096, and round up.
- ▶ Determine the buffer fill and spill (write to the log stream) rate
  - MAXBUF CIs = (MAXBUFSIZE- 40) divided by 4096, round up.
  - Fill rate in seconds = MAXBUF CI divided by # of data CIs.
  - # buffer fills per minute = 60 seconds divided by the fill rate.
- ▶ Define the HIGHOFFLOAD value in CIs
  - HIGHOFFLOAD CIs = MAXBUF CIs times # buffer fills per minute.
- ▶ The size of the staging data set (STG\_SIZE) is:
  - HIGHOFFLOAD CIs times 1.18 - if HIGHOFFLOAD is set to 85%.
  - HIGHOFFLOAD CIs times 1.25 - if HIGHOFFLOAD is set to 80%.

### ***User Journal***

For purposes of discussion, JORC is a CICS transaction which writes to a user journal. It writes 80 byte records and there are 42 writes per task. The transaction rate is 6 transactions per second, running in sequential order.

For a DASD-only log stream, data is written in 4 KB (4096 bytes) control intervals (CI). This means the size of the actual data placed into interim storage is rounded to a 4096 byte boundary.

For DASD-only log streams, MAXBUFSIZE should be set to 65532. With a MAXBUFSIZE of 65532, subtract 40 bytes for the header, leaving 65492 bytes for data. Each record is 148 bytes (80 bytes of data plus 68 bytes of CICS information). Dividing 65492 by 148 yields a record count of 442 will fit in the buffer.

442 records \* 148 bytes/record = 65416 + 40 header = 65456 bytes will be used in each buffer.

Since we know each transaction writes forty-two 148 byte records and the rate is 6 transactions a second, the data written in one second is 42\*148\*6= 37296 bytes. Divide 37296/4096 and rounding up indicates 10 CIs of data are written each second. The buffer holds 16 CIs of data so the buffer will fill every 1.6 seconds or 37.5 times a minute. This equates to 600 CIs/minute (16 CIs per second \* 37.5 fill intervals per minute).

If you want to have an offload occurring roughly once every minute, you must size the data set so that the HIGHOFFLOAD value equates to 600 CIs. For example, if HIGHOFFLOAD is set to 85%, STG\_SIZE would be set to 706 ((100/85) \* 600).

The SMF Type 88 data shown in Figure 5-18 reports the results. The SMF interval has been set to five minutes, note there have been five offloads in the five minute interval. Also note the STG THLD field is also equal to five, indicating there were no IXGWRITE calls issued for this log stream while the offloads were running.

LOGSTREAM NAME--	BYT WRITTN STRUCTURE NAME--	BYT WRITTN BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD	#WRITES INVOKED	AVERAGE			BUFFER SIZE	
						---# WRITES COMPLETED----- TYPE1	TYPE2	TYPE3		
						-----EVENT-----				
						BYT DELETD	# DELETES	BYT DELETD	# DELETES	
						INTERIM ST	W/O DASD	INTERIM ST	W/	OFF- DASD STRC NTRY STG STG RE-
						W/O DASD	WRITE	W/DASD	WRITE	LOAD SHFT FULL FULL THLD FULL BLD
12/02/02 5:25:00 PM (SMF INTERVAL TIMESTAMP 'B89E460005300000'X)										
IYOT1.IY01.DFHJ02	*DASDONLY*	11127520	11837440	11461800	170	0	0	0	65456	
		0	0	12185600	179	<u>5</u>	0	0	<u>5</u>	0 0

Figure 5-18 Sample IXGRPT1 report for a DASD-only User Journal

For a CF-Structure log stream the process is the same, but you must determine how much of the structure will be occupied by the log stream. A point to remember is the data is rounded to a 256 byte boundary when the buffer is written (assuming you used the recommended MAXBUFSIZE of 64000).

With the same transaction rate and record size (42 148-byte records per transaction with a transaction rate of six transactions a second), the data written in one second is 42\*148\*6= 37296 bytes.

Using a CF-Structure log stream which has a MAXBUFSIZE of 64000, 432 records can be written to the buffer ((64000-40)/148).

432 records \* 148 bytes/record = 63936 + 40 header = 63976 bytes will be used in each buffer. The size is then rounded to a 256 byte boundary, making it 64000.



Writing 37296 bytes per second gives a buffer fill rate of 1.7 seconds (64000/37296), or 35.3 times per minute. The amount of data written in one minute is:

$$35.3 \text{ buffers/minute} * 64000 \text{ bytes/buffer} = 2,259,200 \text{ or } 2.2 \text{ MB/minute}$$

Using a HIGHOFFLOAD value of 85%, the portion of the CF structure available to this log stream should have a size of 2.59 MB (2.2 MB \* 1.18) in order to provide a residency time of one minute.

System Logger requires some space for control information, so the size should be rounded up. The structure size should be based on the sum of the calculated values for each stream, plus about 4 MB for System Logger control information (based on CF Level 12).

### ***Auto Journal***

To size a DASD-only auto journal, the process is much the same as described for a User journal. However, in the case of an auto journal CICS will add 84 bytes, plus the length of the key to each block of data. A 40 byte header is also written at the beginning of each block. The sample used in this case is the same transaction (JORC) (without the writes to the user journal). It reads from a VSAM file and rewrites the same record, with JNLUPDATE being specified for the file. The records are 80 bytes with a 6 byte key, there are 30 log writes per task, and the transaction rate is 6 per second.

MAXBUFSIZE is set to 65532 (this is a DASD-only log stream). Subtract 40 bytes for the header, leaving 65492 bytes for data. Each record is 170 bytes (80 bytes of data + 6 byte key + 84 bytes of CICS information). Dividing 65492 by 170, 385 records can be written in each buffer. Therefore, 385 records \* 170 bytes per record + the 40 byte header = 65490 bytes will be used in each buffer.

We know each transaction writes thirty 170-byte records and the rate is 6 transactions a second, so the data written in one second is  $30 * 170 * 6 = 30600$  bytes.

Divide 30600 bytes/second by 4096 bytes/CI and round up for a value of 8 CIs of data. The buffer holds 15 CIs of data (65492/4096) so the buffer will fill every 1.875 seconds or 32 times per minute.

The amount of data written in one minute is calculated as:

$$32 \text{ buffers/minute} * 15 \text{ CIs/buffer} = 480 \text{ CIs per minute}$$

If the desired offload rate is once every minute, the staging data set must be sized so the HIGHOFFLOAD value equates to 480 CIs. For example, if HIGHOFFLOAD is set to 85%, STG\_SIZE would be set to 565  $((100/85) * 480)$ .

Using the value calculated, the SMF Type 88 data shown in Figure 5-19, reports the offloads are happening about every 50 seconds.

LOGSTREAM NAME--	STRUCTURE NAME--	BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD	#WRITES INVOKED	AVERAGE			BUFFER SIZE			
						---# WRITES TYPE1	COMPLETED TYPE2	----- TYPE3				
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----						
		INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-
		W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD
12/01/02 11:15:00 PM (SMF INTERVAL TIMESTAMP 'B89D525DB850000'X)												
IYOT1.IY01.DFHJ03	*DASDONLY*	12901530	13717504	13761300	197	0	0	0	0	0	65490	
		0	0	14622720	214	<u>6</u>	0	0	0	<u>6</u>	0	0

Figure 5-19 Sample IXGRPT1 report for a DASD-only Auto Journal log stream

For a CF-Structure log stream, the process is the same, but you must determine how much of the structure will be occupied by the log stream. Another point to remember is the data is rounded to a 256 byte boundary when the buffer is written (assuming you used the recommended MAXBUFSIZE of 64000).

With the same transaction rate and record size (thirty170 byte records per transaction with a transaction rate of 6 transactions a second), the data written in one second is  $30 \times 170 \times 6 = 30600$  bytes. A 40 byte header is added at the start of each buffer.

Using a CF-Structure log stream which has a MAXBUFSIZE of 64000, 376 records can be appended to each buffer  $((64000-40)/170)$ .

$376 \text{ records} \times 170 \text{ bytes/record} = 63920 + 40 \text{ header} = 63960$  bytes will be used in each buffer. The size is then rounded to a 256 byte boundary, making it 64000.

Writing 30600 bytes per second gives a buffer fill rate of once every 2.0 seconds  $(64000/30600)$  or 30 times per minute.

The amount of data written in one minute therefore is:

$$30 \text{ buffers/minute} \times 64000 \text{ bytes/buffer} = 1,920,000 \text{ (1.92M) bytes/minute.}$$

Using a HIGHOFFLOAD value of 85%, the portion of the CF structure available to this log stream should have a size of 2.27 MB  $(1.92 \text{ MB} \times 1.18)$  in order to provide a residency time of one minute.

System Logger requires some space for control information, so the size should be rounded up. The structure size should be based on the sum of the calculated values for each stream, plus about 4 MB for System Logger control information (based on CF Level 12).

### Forward Recovery Logs

To size a DASD-only forward recovery log, the process is once again the same with the difference in the fact records are forced out (written to the log stream) at sync point. In addition, the record volumes are usually higher because a single log stream is often used for many files.

Once again, the JORC transaction is used with a transaction rate of 6 per second. Since the records are forced at sync point, the number of records logged in the UOW must be understood. In the sample, the transaction issues 7 read/updates followed by a rewrite in each UOW. The records logged are 80 bytes in length with a 6 byte key. There are multiple

UOWs per task, which does not affect the result, since the buffer is written during sync point processing for each UOW.

Each record is 170 bytes (80 bytes of data + 6 byte key + 84 bytes of CICS information). The amount of data written for each UOW, is (7 records per UOW) \* (170 bytes per record) + 40 (header) = 1230 bytes. So even though MAXBUFSIZE is set to 65532, the actual buffer space used is only 1230 bytes per UOW. There are 6 UOWs per task. With a transaction rate of 6 transactions per second, each writing 6 buffers (1 for each UOW), there are a total of 36 1230-byte buffer writes per second (44,280 bytes).

For DASD-only, each buffer write requires a 4 KB CI, even though there are only 1230 bytes. Therefore, 36 CIs are used each second, 2160 (36\*60) are used in a minute. In this case, 2160 CIs times 1.18 (assuming HIGHOFFLOAD of 85%) = a STG\_SIZE of 2549 for a residency time of one minute.

Using an STG\_SIZE of 2549 CIs, the SMF Type 88 data Figure 5-20 shows 5 offloads in the 5 minute SMF interval. Note the number of staging thresholds is 55, indicating an average of 10 log writes were performed during each of the offload periods. Ten isn't anything to worry about; however, if the number begins to grow it may be an indication of problems in the offload processing, such as allocation of a new offload data set. Another option would be to increase the STG\_SIZE slightly to allow more data to be written between each offload.

BYT WRITTN			BY USERS		TO INTERIM	TO DASD	AVERAGE				BUFFER		
LOGSTREAM	NAME--	STRUCTURE	NAME--	IXGWRTES	STORAGE		#WRITES	---#	WRITES	COMPLETED-----	SIZE		
							INVOKED	TYPE1	TYPE2	TYPE3			
			BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----						
			INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-
			W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD
12/02/02 8:25:00 PM (SMF INTERVAL TIMESTAMP 'B89E6E3BB3F00000'X)													
IYOT1.IY01.DFHJ03		*DASDONLY*	11729280	39059456	12593320	9536	0	0	0	0	1230		
			0	0	37609472	9182	<u>5</u>	0	0	0	<u>55</u>	0	0

Figure 5-20 Sample IXGRPT1 report for a DASD-only Forward recovery log

For a CF-Structure log stream, the process is the same, but you must determine what percentage of the structure will be occupied by the log stream. Also remember the data is rounded to a 256 byte boundary when the buffer is written (assuming the recommended MAXBUFSIZE of 64000 is used).

The transaction rate and record size (six 1230-byte records per transaction with a transaction rate of six per second) remain the same.

Using a CF-Structure log stream which has a MAXBUFSIZE of 64000, the size of the data written (1230) is rounded to a 256 byte boundary, 1280 bytes.

There will be 36 buffers written in a one minute interval (6 per transaction and 6 transactions per second), so the data written is:

$$1280 \text{ bytes/buffer} * 36 \text{ buffers/minute} = 46080 \text{ bytes per minute.}$$

Using a HIGHOFFLOAD value of 85%, the portion of the CF structure available to this log stream should have a size of 55 KB (46080 \* 1.18) to provide a residency time of one minute.

System Logger requires some space for control information, so the size should be rounded up. The structure size should be based on the sum of the calculated values for each stream, plus about 4 MB for System Logger control information (based on CF Level 12).

**Attention:** A common problem is the *over*-sizing of general log streams (user journals, forward recovery log streams, and auto journals). Oversizing causes greater amounts of data to be retained in interim storage for a longer period of time. This leads to an increase in the working set (page frames) associated with the System Logger data space. In addition, the length of time required to offload the data is increased.

### **DFHLSCU - Log Stream Sizing utility**

This section describes the use of DFHLSCU to analyze pre-CICS Transaction Server journals to determine the specifications for CICS Transaction Server log streams. If pre-CICS Transaction Server journals are not available, refer to “When DFHLSCU is not used for sizing” on page 177.

When migrating to CICS Transaction Server from an older release of CICS, the Log Stream sizing migration utility (DFHLSCU), provided in CICS Transaction Server, should be run against the CICS V4.1 system journal (DFHJ01). It's best to use the journal from a busy day in order to size for the peak workload.

DFHLSCU divides the CICS V4.1 journal into time segments based on the INTERVAL parameter supplied as input for the job. It then analyzes the log records found in each segment to determine the busiest segment based on the logging activity. The busiest segment is then used to determine the parameters for your log stream definitions and space requirements.

DFHLSCU reads the CICS V4.1 journal and converts the record sizes to the equivalent values for CICS Transaction Server. Multiplying the number of each record type by the record sizes for CICS Transaction Server provides the amount of data which will be written in a given interval.

The IXGIZE service of the System Logger is then called to provide a size estimate for the interim storage portion of the log stream.

A value of 0 should *not* be specified for the INTERVAL value. A value of 0 indicates all records on the journal are to be considered as written in the same interval, that is, no time-segmenting is to occur and the period covered by the entire data set is to be used. The result will be a recommendation for a very large interim storage for the log stream.

When using DFHLSCU to define the size for DFHLOG, the value specified for TRANSDUR should reflect the length of time between sync points in the applications. The default for TRANSDUR is 3 seconds; remember *this is not transaction response time* unless the transaction does not issue sync points, that is, the transaction contains only one unit of work.

The actual value for the transactions in the region being migrated should be used.

If the transactions consist of a single UOW, the response time may be used.

The control statements for DFHLSCU are:

- ▶ AKPFREQ is the value specified in the System Initialization Table (SIT)

The current setting can be verified using the master terminal command CEMT I SYSTEM.

- ▶ Calculated values:
  - The number of log writes per transaction
 

The number of log writes per transaction can be estimated using the CICS performance records, or by dividing the total journal writes by the number of transactions. For CICS V4.1:

$$\text{AKPINTVL} = \text{AKPFREQ} / (\text{the number of trans per second} * \text{log writes per transaction})$$

$$\text{Number of list entries (no. entries)} = ((\text{akpintvl} + \text{trandur}) * \text{writes per sec}) / 0.9$$
- ▶ TRANSDUR - With the assumption each transaction consists of a single UOW, this is the average response time. If the transactions consist of multiple UOWs, the length of the longest UOW should be used.
- ▶ TRANSACTION Rate or the number of transactions processed per second. The number of transactions can be determined using DFH0STAT, the CICS shutdown statistics, or summarizing the CICS performance records for the interval. Divide by the length of the interval to determine the transactions per second.

Refer to the manual *CICS TS for z/OS Operatins and Utilities*, SC34-5991 for further details.

**Tip:** The transaction rate and response time may be obtained by defining an RMF Report Class for each group of transactions or for all transactions within a region. The report in Figure 5-21 shows the transaction rate is 34.61 trans/sec with a response time of 0.259 seconds. Report Classes can be defined for CICS V4.1 as well CICS Transaction Server.

Refer to “Defining Service Classes and Performance Goals” in *z/SO MVS Planning: Workload Management*, SA22-7602.

REPORT BY: POLICY=WLPOL		REPORT CLASS=RJRRY1	
DESCRIPTION =Report class for the RJRR transactions			
TRANSACTIONS	TRANS.-TIME	HHH.MM.SS.TTT	
AVG 0.00	ACTUAL	259	
MPL 0.00	EXECUTION	0	
ENDED 10383	QUEUED	0	
END/S 34.61	R/S AFFINITY	0	
#SWAPS 0	INELIGIBLE	0	
EXCTD 0	CONVERSION	0	
AVG ENC 0.00	STD DEV	431	
REM ENC 0.00			
MS ENC 0.00			

Figure 5-21 Sample Report Class output

### 5.3.6 Performance

Obtaining the best performance is a matter of tuning for the optimum data residency time in interim storage. For CICS system log streams (DFHLOG and DFHSHUNT), data required for backout and recovery should be available in interim storage for the duration of the unit of work. For non-system log streams, the data residency time should be such that a steady movement of data from interim storage to the offload data sets is observed while requiring a minimal amount of storage to duplex the data within the System Logger data space.

The performance of CICS log streams is influenced by the sizing, type, and placement of the log streams. In addition, activity keypoint frequency (AKPFREQ), and the log defer interval

(LGDFINT) can have a significant impact on log stream performance. For information about parameter specification, refer to:

- ▶ “AKPFREQ” on page 153
- ▶ “LGDFINT” on page 154
- ▶ “Define the log stream in the LOGR policy” on page 159
- ▶ “DASD-only log stream” on page 164

It's important to remember the IXGWRITE calls are issued under the CICS Quasi-Reentrant (QR) TCB, hence the CPU time is charged to the transaction.

## Performance tools

Performance data is available from several sources:

SMF Type 88 records produced by the z/OS System Logger.

SMF Type 110 data produced by CICS.

Includes both statistical and performance data.

SMF Type 70-78 data produced by RMF (or an equivalent product).

There are several tools which aide in the analysis of log stream performance problems:

- ▶ DFH0STAT - a CICS-supplied sample transaction (STAT) collects the CICS statistics and writes them to the CICS JES log. Refer to “DFH0STAT” on page 188 for details.
- ▶ DFHSTUP (CICS Statistics Utility Program) - used to post-process the CICS SMF Type 110 sub type 2 statistical records. Refer to “DFHSTUP” on page 192 for details.
- ▶ CICS Performance Analyzer (part of the CICS Tools portfolio) - used to post-process the CICS SMF 110 sub type 1 performance records and SMF Type 88 records. Refer to “CICS Performance Analyzer” on page 193 for details.
- ▶ IXGRPT1 (sample program provided with z/OS) formats and reports the SMF Type 88 data produced by the z/OS System Logger. Refer to “SMF Type 88 records and IXGRPT1 program” on page 291 for an explanation of the report fields. Refer to “IXGRPT1 observations and possible actions” on page 195 for a description of the fields which are interesting in a CICS environment.
- ▶ Various RMF reports (Coupling Facility Activity reports, Workload Activity reports) based on the data captured in the SMF 70 -78 records. Refer to “RMF Coupling Facility Activity Report” on page 287 for details.
- ▶ The sample DFSORT jobs provided with this book in 8.6.5, “DFSORT jobs to format SMF 88 records” on page 297. These reports provide similar information to the IXGRPT1 program, however they allow you to summarize all the activity by log stream rather than by interval, and they are also more readable than the IXGRPT1 output.

### **DFH0STAT**

(Use: Dynamically gather CICS statistics)

DFH0STAT is supplied as a sample COBOL program in CICS.SDFHSAMP. It contains self-documenting source code which can be compiled and run as a transaction to collect CICS statistics and write them to the JES spool. The output can be viewed under TSO. The SIT parm SPOOL=YES is required. The information presented with DFH0STAT is also available in the CICS shutdown statistics. The interval reported is based on the CICS stats interval—to be precise, it is actually the data since the last stats reset, which is determined by the stats interval.

A compiled version is available in CICS Transaction Server V2.2.

A sample output is shown in Figure 5-22. The MVS release is provided in the System Status section, along with the values for Activity Keypoint Frequency (AKPFREQ) and log stream Deferred Force Interval (LGDFINT).

System Status											
MVS Product Name . . . . : MVS/SP6.1.0											
Activity Keypoint Frequency . . . . . : 4,000											
Logstream Deferred Force Interval . . . . : 30											
Logstream Name	Use count	Status	Sys Log	Structure Name	Max Block Length	DASD Only	Retention Period	Auto Delete	Stream Deletes	Browse Starts	Browse Reads
IYOT1.DFHLOG	1	OK	YES		65,532	YES	0	NO	3	34	0
IYOT1.DFHSHUNT	1	OK	YES	LOG_JG	64,000	NO	0	NO	1	0	0
IYOT1.J02	1	OK	NO		65,532	YES	0	NO	N/A	N/A	N/A
Logstream Name	Write Requests	Bytes Written	Average Bytes	Buffer Appends	Buffer Full Waits	Force Waits	Current Waiters	Peak Waiters	Retry Errors		
IYOT1.DFHLOG	9,023	2,546,491	282	9,037	0	2	0	1	1		
IYOT1.DFHSHUNT	0	0	0	0	0	0	0	0	0		
IYOT1.J02	14	916,800	65,485	9,009	0	0	0	0	0		

Figure 5-22 Sample DFH0STAT output

In the log stream statistics we see each log stream connected to this CICS region. If the log stream resides in a CF structure, the structure name is given; if it is a DASD-only log stream, the structure name is blank. In the example, IYOT1.DFHLOG and user journal J02 are DASD-only log streams, while IYOT1.DFHSHUNT is a CF-Structure log stream in structure LOG\_JG.

The values under Max Block Length are worth noting. This value is specified in the log stream or structure definition and is returned to CICS when it connects to the log stream. For a CF-Structure log stream the blocksize is specified as MAXBUFSIZE on the structure definition. The value specified in MAXBUFSIZE determines the element size for the log streams in the structure. If the MAXBUFSIZE is specified equal to or less than 65276, the element size is 256; if greater than 65276 the element size is set to 512.

For DASD-only log streams, MAXBUFSIZE may be specified on the log stream definition. MAXBUFSIZE defines the largest block which can be written to the log stream. The default value is 65532.

In either case, the MAXBUFSIZE value is returned to CICS and is used to determine the CICS log stream buffer size. Note, for user journals, unless the application uses the WAIT option, the IXGWRITE call is issued when the buffer fills. Refer to the Average Bytes column. This is the average buffer size written using IXGWRITE calls. For DFHJ02 the average size is 65,485. It is sometimes advantageous to reduce MAXBUFSIZE on a user journal to force the data into the log stream more frequently. This reduces the residency time and allows quicker access to the data from batch applications.

It's also worth noting in the case of a hard failure (MVS, CICS, or the hardware) where the data has not be written to the log stream, it will be lost. For DFHLOG and DFHSHUNT, when the region is emergency restarted, in-flight tasks would be backed-out using the information which had already been written to the log.

In the case of a user journal, the data in the buffer would be lost. There is no transaction backout associated with data lost in a non-system log buffer.

The columns entitled DASD Only, Retention Period, and Auto Delete reflect the attributes specified for the log stream in the System Logger policy.

The value given under Stream Deletes is the number of times CICS issued an IXGDELETE call to the System Logger for log tail deletion.

The value under Browse Starts is a count of the number of times a browse start request is issued. Prior to CICS Transaction Server V2.2, a large value may be observed for DFHLOG in a low volume region, due to CICS using a Browse Start to verify the System Logger is still operational. In CICS Transaction Server V2.2, a change was implemented to call the System Logger function CHECK\_CONNECTION\_STATUS and as a result this number may not be as large as in a CICS Transaction Server V2.2 system.

The Browse Reads column reflects the number of times CICS reads the log stream to obtain information for transaction backout.

The number of Write Requests is the number of times CICS called the System Logger via an IXGWRITE. The number of Buffer Appends is a count of the times a record was added (appended) to the CICS log buffer. The number is normally larger than the number of Write Requests due to calls to the CICS logger domain which do not include the force option. For example, the records produced during activity keypoint processing are not written with the force option. In contrast, the before images for a recoverable VSAM file are written with the force option specified. Unfortunately it is not possible to predict what percentage of writes will be with the force option, it depends on the work load. If the workload is comprised of mostly updates for a recoverable VSAM file, most of the records on DFHLOG will be forced. If the bulk of the records are the result of an activity keypoint we simply append the records without force.

Buffer Full waits is a count of the number of attempts to append a record to the current log stream buffer while the buffers were logically full. This situation arises when the current log stream buffer does not have sufficient space to accommodate the journal record, and the IXGWRITE is still in progress for the alternate log stream buffer.

Force Waits is the total number of tasks suspended waiting for an IXGWRITE call to complete.

Buffer Full Waits and Force Waits can be an indication there is a delay in I/O processing. They can also be an indication the log defer interval is too large. Additional items to check are the CF service times for a Coupling Facility log stream or DASD I/O time for DASD-only log streams.

If Buffer Full and Force waits are seen consistently, verify the value for LGDFINT is set to 5. If the Task Journal Wait Time (JCIOWTT) is very close to the LGDFINT value, then the LGDFINT value should be decreased. Prior to CICS Transaction Server V2.2, the default value was 30. Normally the recommendation is not to reduce the value lower than 5, however, in a very high volume region it may be advantageous to set as low as 0 for CF-Structure log streams. In addition, CF service time (for CF-Structure log streams) or DASD I/O time should be investigated.

Setting LGDFINT to zero causes the data to be written to the log stream (via an IXGWRITE) immediately without waiting for additional records to be added to the buffer.

The Peak Waiter field contains the high water mark of the number of tasks (UOWs) waiting for a buffer to be written. The Current Waiter field is the number of tasks (UOWs) waiting for a buffer to be written at the point in time the statistics were collected.



Retry Errors is the number of requests to System Logger resulting in a return code indicating an error, which was subsequently retried. An example would be 868 errors returned while the staging data set is being formatted. This can happen with DASD-only log streams or if staging data sets are used with a CF-Structure log stream. For example:

```
01.48.56 JOB07716 +DFHLOG0777 IY0T1
A temporary error condition occurred during MVS logger operation IXGWRITE for log
stream IY0T1.DFHLOG. MVS logger codes: X'00000008', X'00000868'.
```

With CICS Transaction Server V2.2, DFH0STAT has been enhanced. The format has been changed for ease of reading and information has been added to calculate (and display) the number of log stream writes per second. The samples in Figure 5-23 and Figure 5-24 on page 192 show the format of the new reports.

Applid IY0T1	Sysid IY01	Jobname IY0T122	Date 08/11/2002	Time 13:50:13	CICS 6.2.0
<b>System Status</b>					
MVS Product Name . . . . .	MVS/SP7.0.4			CICS Transaction Server Level . . .	02.02.00
CICS Startup . . . . .	INITIAL			RLS Status . . . . .	RLS=NO
CICS Status . . . . .	ACTIVE			RRMS/MVS Status . . . . .	OPEN
Storage Protection . . . . .	ACTIVE			VTAM Open Status . . . . .	OPEN
Transaction Isolation . . . . .	ACTIVE			IRC Status . . . . .	OPEN
Reentrant Programs . . . . .	NOPROTECT			TCP/IP Status . . . . .	OPEN
Force Quasi-Reentrant . . . . .	No			Max IP Sockets . . . . .	255
Program Autoinstall . . . . .	ACTIVE			Active IP Sockets . . . . .	0
Terminal Autoinstall . . . . .	ENABLED			WEB Garbage Collection Interval . .	60
Activity Keypoint Frequency . . . . .	1,000			Terminal Input Timeout Interval . .	5
Logstream Deferred Force Interval . . . . .	0				
DB2 Connection Name . . . . .	SYSD				

Figure 5-23 DFH0STAT for CICS TS 22 - part 1

Applid	Sysid	Jobname	Date	Time	CICS	PAGE					
IYOT1	IY01	IYOT122	08/11/2002	13:50:13	CICS 6.2.0	10					
Logstream - System Logs											
System log - DFHLOG											
Logstream Name	IYOT1.CICS22.DFHLOG			Logstream Status	OK						
DASD Only	YES			Retention Period (days)	0						
Coupling Facility Structure Name				Auto Delete	NO						
Logstream Writes	18,788			Maximum Block Length	65,532						
<u>Logstream Writes per second</u>	<u>14.74</u>										
Average Bytes per Logstream Write	1,289										
Logstream Deletes (Tail Trims)	105										
Logstream Query Requests	3										
Logstream Browse Starts	0										
Logstream Browse Reads	0										
Logstream Buffer Appends	104,811										
Logstream Buffer Full Waits	0										
Logstream Force Waits	93,764			Logstream Current Force Waiters	3						
Logstream Retry Errors	1			Logstream Peak Force Waiters	11						
-----											
Applid	Sysid	Jobname	Date	Time	CICS	PAGE					
IYOT1	IY01	IYOT122	08/12/2002	08:31:13	CICS 6.2.0	11					
Logstreams - Resource											
Logstream Name	Use Count	Status	Sys Log	Structure Name	Max Block Length	DASD Only	Retention Period	Auto Delete	Stream Deletes	Browse Starts	Browse Reads
IYOT1.CICS22.DFHLOG	1	OK	YES	LOG_JG_20M	64,000	NO	0	NO	1,896	0	0
IYOT1.CICS22.DFHSHUNT	1	OK	YES	LOG_JG_20M	64,000	NO	0	NO	1	0	0
Logstreams - Requests											
Logstream Name	Write Requests	Bytes Written	Average Bytes	Buffer Appends	Buffer Full Waits	Force Waits	Current Waiters	Peak Waiters	Retry Errors		
IYOT1.CICS22.DFHLOG	445,574	443,483,499	995	1,895,148	0	428,880	1	10	1		
IYOT1.CICS22.DFHSHUNT	0	0	0	0	0	0	0	0	0		

Figure 5-24 DFH0STAT for CICS TS 2.2 - part 2

### DFHSTUP

(Use: Evaluate the CICS statistics on a CICS statistics interval or daily basis)

The CICS statistics utility program, DFHSTUP, is an offline utility which processes CICS statistics information contained in SMF Type 110 subtype 2 records. The CICS statistics domain records interval statistics when the system initialization parameter STATRCD=ON is specified. Other statistics record types (unsolicited, requested, and end-of-day) are written regardless of the setting of the STATRCD option. For additional information about DFHSTUP refer to *CICS TS for z/OS Operations and Utilities*, SC34-5991.

A separate version of DFHSTUP is provided with each release of CICS, so the correct version must be used to process the data.

In the sample DFHSTUP EOD (End of Day) report shown in Figure 5-25, you will notice the same type of information provided by DFH0STAT is presented. However, in this case it is for the entire execution of the CICS job, or for a 24 hour period if the CICS region was executing for multiple days.

Journal Name			Journal Type	Log Stream Name	Write Requests	Bytes Written	Buffer Flushes
DFHLOG	MVS	IYOT3.DFHLOG			N/A	N/A	N/A
DFHSHUNT	MVS	IYOT3.DFHSHUNT			N/A	N/A	N/A
*TOTALS*					0	0	0

Log Stream Name	System Log	Structure Name	Max Block Length	DASD Only	Retention Period	Auto Delete	Delete Requests
IYOT3.DFHLOG	Yes	LOG_JG2_20M	60000	No	0	No	45
IYOT3.DFHSHUNT	Yes	LOG_JG_20M	64000	No	0	No	1
*TOTALS*							46

Log Stream Name	Write Requests	Bytes Written	Buffer Appends	Waits Buff	Current Frce Full	Peak Frce Wtrs	Total Force Wts	Browse Starts	Browse Reads	Retry Errors
IYOT3.DFHLOG	90148	45213095	178839	0	0	3	554	46	111	0
IYOT3.DFHSHUNT	0	0	0	0	0	0	0	0	0	0
*TOTALS*							554	46	111	0

Figure 5-25 Sample DFHSTUP End of Day report

In the sample report, notice the log stream IYOT3.DFHLOG has been specified with a MAXBUFSIZE of 60000, is allocated to structure LOG\_JG2\_20M, has a retention period of 0, and auto delete is set to NO.

The EOD report shows the overall summary for the day. To collect interval statistics, the interval may be set using the STATINT= system initialization parm, or it can be set via the CEMT master terminal transaction.

### CICS Performance Analyzer

(Use: Analysis of the SMF 110 Subtype 1 CICS Performance Records)

CICS PA is a post processor for CICS SMF Type 110 subtype 1 records. It provides the ability to summarize by CICS monitoring fields, and display the API calls and resources used, along with CPU per transaction and response time.

In Release 2 of CICS Performance Analyzer, the product was enhanced to provide support for the System Logger SMF Type 88, and SMF Type 101 records produced by DB2®. The

sample outputs from IXGRPT1 in Figure 5-27 and from CICS PA in Figure 5-26 show the different methods for displaying the data.

VIR2M0		CICS Performance Analyzer System Logger - Logstream Summary							
LOGR0001 Printed at 11:15:59 5/28/2003		Data from 17:20:00:00 5/27/2003 to 17:20:00:00 5/27/2003							
Logger data									
Logstream name	MVSID	Structure name	First interval start	Last interval stop	Total Interval				
IYOT1.CICS22.DFHLOG	SYSD	LOG_JG2_5M	17:15:00.00 5/27/2003	17:20:00.00 5/27/2003	0000:05:00				
----- IXWRITES -----			----- DELETIONS -----						
			Bytes	Count	Count	Bytes	Bytes		
			Writn to	With	Without	After	Int Stor		
			Interim	DASD	DASD	Offload	w/o DASD		
	Count	Total	Storage	Write	Write	w. DASD	Write		
		Average							
		Bytes							
Total	1138714	324227K	285	556595K	103114	1067720	30936K	302169K	
Rate(/Sec)	3795	1080756		1855317	343	3559	103119	1007232	
Minimum	1138714	324227K		556595K	103114	1067720	30936K	302169K	
Maximum	1138714	324227K		556595K	103114	1067720	30936K	302169K	
----- EVENTS -----									
		Demand					Demand		
		DASD	Block	Staging	Entry	Struct	Init'd		
	Offloads	Shifts	Length	Full	Full	Full	Offloads		
		Staging							
		Threshld							
Total	378	0	5	0	3470	5	0		
Rate(/Sec)	1	0	0	0	11	0	0		
Minimum	378	0	5	116	0	3470	5	0	
Maximum	378	3470	5	3308	0	3470	5	0	
----- EVENTS -----					----- DASD Writes -----				
			Struct	Struct		Total			
			Rebuilds	Rebuilds	Count	Bytes	Average	Waits	
	Type1	Type2	Init'd	Complt'd					
Total	1072720	46589	19401	0	0	323	35060K	0	236
Rate(/Sec)	3575	155	64	0	0	1	116867		0
Minimum	1072720	46589	19401	0	0	323	35060K		236
Maximum	1072720	46589	19401	0	0	323	35060K		236

Figure 5-26 CICS Performance Analyzer Log stream Summary

BYT WRITTN	BYT WRITTN	BYT WRITTN	AVERAGE		---# WRITES COMPLETED---						BUFFER		
LOGSTREAM NAME----	STRUCTURE NAME-	IXGWrites	BY USERS	TO INTERIM	TO DASD	#WRITES	TYPE1	TYPE2	TYPE3	SIZE			
			TO INTERIM	STORAGE	STORAGE	INVOKED							
			BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----						
			INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-
			W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD
IYOT1.CICS22.DFHLOG	LOG_JG2_5M	324226786	556595200	35060218	1138714	1072720	46589	19401	284				
		302169464	1067720	30935658	103114	378	5	5	3470	0	0	0	

Figure 5-27 Sample IXGRPT1 Report

## ***IXGRPT1 observations and possible actions***

The following fields should be noted in the IXGRPT1 reports for CICS log streams:

1. For DFHLOG and DFHSHUNT, the number of “BYT DELETED INTERIM ST W/O DASD” should be very close to the “BYT WRITTEN TO INTERIM STORAGE”. A non-zero value in “BYT DELETED INTERIM ST W/DASD” indicates data is being offloaded and then deleted, costing additional processing and I/O. The BYT WRITTEN TO DASD should be very low.

Factors to be considered include:

- There are long running CICS tasks, preventing log tail deletion.

This is not simply CICS transactions with high response times, but rather long running tasks which use recoverable resources and do not issue a sync point. In addition, the tasks are causing log records to be written during each activity keypoint interval.

If message DFHLG0743 (Tail of log stream log\_stream\_name deleted at block id X'nnnn') is not being issued for DFHLOG with each activity keypoint (message DFHRM0205), a long running task is preventing tail trimming. However, it is not unusual to see infrequent DFHLG0743 messages for DFHSHUNT. Units of work may exist on DFHSHUNT for extended periods of time. Examples are conversational tasks which have updated a recoverable resource, and mirror tasks awaiting a forget flow from the connected region.

In CICS Transaction Server, the forget flow (part of 2 phase commit) is carried with the next task attach to flow across the MRO link. This improves performance on most links, but if the usage is low, the log records for the mirror may reside on DFHSHUNT for an extended period of time. A DFHLG0743 message being issued for DFHSHUNT indicates units of work, which had been inactive, have completed. CICS APARs PQ22563 and PQ14796 reduce the amount of data recorded on DFHSHUNT. In addition, APARs PQ56315 (1.3) and PQ56341 (2.2) should be applied.

- AKPFREQ is set too high

Use the DFHRM0205 messages to determine how often an activity keypoint is being taken.

- Problems related to interim storage

- CF-Structure log stream structure size is too small, or staging data set cannot hold as much data as the log stream structure storage.
- Incorrect entry-to-element ratio—this can happen when log streams with very different block sizes or very different write rates are in the same structure. Dynamic changes to the entry-to-element ratio will not be made more frequently than once every 30 minutes.
- DASD-only log streams—the size of the staging data set may be insufficient.

- HIGHOFFLOAD

Should be set no higher than 85%.

- LOWOFFLOAD

- For DFHLOG and DFHSHUNT, set in the range of 40 - 60%.
- For user journals, set to 0%. All data should be offloaded each time an offload is initiated.

2. Under # WRITES COMPLETED (prior to the installation of the PTF for APAR OA03055, this is only provided for CF-Structure log streams):

- TYPE1 -- normal - this number should be high.
- TYPE2 -- normal, but the log stream reached or exceeded the HIGHOFFLOAD value.
- TYPE3 -- writes issued when 90% of the elements for the log stream are in use.

- Look for HIGHOFFLOAD set to 90% or higher.
- Tail trimming is not happening or is not being effective (see item 1 on page 195).
- CICS is filling the space above HIGHOFFLOAD point faster than the log stream is being offloaded.

### 3. Under EVENTS

- NTRY FULL indicates the number of times *all* log streams in the structure were offloaded due to reaching 90% of the structure entries in use.

This could be the result of the entry-to-element ratio being too large (AVGBUFSIZE is specified larger than necessary), or a poorly behaving application which is causing many small records to be written to a log stream which normally contains large records.

- OFFLOADs are good if they are triggered by the HIGHOFFLOAD value. However, offloads are bad if they are triggered by an NTRY FULL condition. In addition, for a CF-Structure log stream, offloads should be triggered because the CF-Structure log stream reached its HIGHOFFLOAD value, *not* because the staging data set reached its HIGHOFFLOAD point (see STG THLD below).
- DASD Shifts indicates the number of times a new offload data set is allocated.
  - For DFHLOG and DFHSHUNT this number should be very small. A large value here indicates too much data is being offloaded (see item 1 on page 195).
  - For user journals, each offload data set should be capable of holding multiple offloads of the log stream.
  - If the offload data set size has not been specified in the log stream definition (LS\_SIZE) or in the SMS data class, the size will be determined by the value specified in the ALLOCxx member of SYS1.PARMLIB -- which defaults to 2 tracks.
- STRC Full indicates the number of times a structure full condition was reached—this should always be 0.
- STG THLD is the number of times the HIGHOFFLOAD percentage was reached for the staging data set.
 

This is good for DASD-only log streams but should not happen for CF-Structure log streams. If non-zero values are seen here for a CF-Structure log stream, the staging data set size needs to be increased so it will hold at least as much data as the log stream in the structure. A value here for a CF-Structure log stream also indicates the CF is either volatile or failure dependent or DUPLEXMODE(UNCOND) has been specified for the log stream.
- Rebuild indicates the number of structure rebuilds in the interval—if this happens on a regular basis, it needs investigation.

## Performance Analysis

Following are considerations about performance for CICS log streams.

### System Log Streams

The sample discussed in “Sample workload and methodology” on page 198 shows different runs of a workload with different system set up to show how the change in these values can influence the CICS and System Logger environment with the results from the different runs. If you need additional details from a benchmark perspective of all the different and additional runs and configurations, refer to the Redpaper™ REDP-3768-00.

## ***Log Stream Characteristics and Placement***

In addition to tuning the CICS and system logger, there are some tuning considerations that need to be evaluated on the surrounding environment, for example the log stream placement and the DASD pooling are very important factors.

The placement and size of the log stream(s) you wish to tune must be understood. For example, are the staging and offload data sets allocated to DASD pools which provide the best performance? This not only includes the actual device access time, but activity which could make the device unavailable. For example, a reserve from another system or high activity data sets which could result in the request being queued.

Is there possible contention between log streams? For example, when there are multiple log streams in a given structure, do they have similar characteristics, that is, the volume and block size of the data written?

If the log stream in question resides in a CF, determine the largest average buffer size for log streams in the structure. This value is used to determine the entry-to-element ratio. The element size is determined by the MAXBUFSIZE value specified when the structure is defined in the System Logger policy (using IXCMIAPU). When the value is equal or less than 65276, the element size is 256. If the value specified is greater than 65276, the element size is 512.

For example, if the element size is 256 and the IXGWRITE is for 734 bytes, 3 elements are required.

The entry-to-element ratio is dynamically adjusted based on a current snapshot of the usage of all log streams connected to the structure. The snapshot will be taken about every 30 minutes and the ratio adjusted if necessary. For more information, refer to “The entry-to-element ratio” on page 26.

If there are 3 log streams in the structure, with average buffer sizes of 1800, 1450, and 734, System Logger will use the current real time average (how many records of each size) to define the entry-to-element ratio. Assuming an element size of 256 bytes, if most of the records are in the 1800-byte range the average would be 1:8. This means we expect to use 8 elements for each entry.

Consider the case where the application writes many short records, say 200 bytes, to the same log stream resulting in an average size of 734. In this situation, one entry is still used for each write, but only one of the eight assumed elements is used. The net effect is more entries are used than predicted leading to an entry full condition before HIGHOFFLOAD can be reached. Offload will be triggered, but there is no extra room to write additional records until some space is recovered. In this situation the NTRY FULL count will be equal to or greater than the number of offloads.

The important point to remember is a log stream performs best when it is contained in a structure where all log streams have similar characteristics (that is, average buffer size and amount of data written). Typically, the log streams for TORs, AORs, and FORs are different and should not be mixed. Even among AORs, two AORs may have completely different log stream characteristics. In addition, DFHLOG and DFHSHUNT should not be placed in the same structure.

Prior to z/OS 1.2, CF writes greater than 4 KB are written asynchronously which can increase response time. However, z/OS 1.2 introduced a new heuristic algorithm for deciding whether a given CF request should be sent synchronously or asynchronously. This new algorithm is discussed in Washington System Center Flash10159. The important point is that System Logger knows whether a given IXGWRITE request will be processed synchronously or

asynchronously—for synchronous requests, it does not respond to the IXGWRITE requestor until the command completes, whereas for asynchronous writes, it responds with a return code indicating the write was asynchronous, and the result will be POSTed later.

**CF Hardware**

It's very important to understand if the log streams will be in a structure which resides in a standalone CF or in a CF which resides in the same CPC as one or more of the connected systems.

A standalone CF is considered volatile when it no longer has a battery backup—that is, in the event of a power loss, the data could not be preserved. If the CF is in the same physical CPC as the z/OS LPAR, it is considered to be failure dependent, or in the same failure domain. In either case, System Logger will (if STG\_DUPLEX(YES) is specified for that log stream) duplex the data to the staging data set rather than its data space. This ensures data integrity, but increases the cost and response time for log writes.

A symptom of this condition is when the CICS address space is initializing and messages are received indicating a staging data set is being formatted. Refer to “STG\_SIZE” on page 162 for considerations during log stream definition.

**Sample workload and methodology**

The data collected includes the SMF Type 88 records, SMF Type 70 to 79 records, and the SMF Type 110 (CICS) records. The data was combined in the following charts by sample.

The workload consists of up to three CICS regions (CICS Transaction Server V2.2 and V1.3) executing under z/OS R1.4.

For the test runs, a transaction (JORM) was started via terminal input. The application reads and writes 7 records from/to a VSAM file (FILEA), issues a sync point, then repeats the process 6 times. It then issues an EXEC CICS Start for ten transactions (JORA-J). Each of these transactions reads and writes 7 records from/to a unique VSAM (FILEA-J), issues a sync point and repeats the process six times. It will then issue an EXEC CICS START for itself. This will repeat up to the number of repetitions passed from JORM (x'7500'). The 10 tasks run in parallel for 30 minutes and the region is then canceled. In cases where 3 regions are used, the process is the same in each region.

The only log streams that we are interested in these sample study are DFHLOG and DFHSHUNT. The log streams are varied between a CF-Structure log stream and DASD-only. No data is ever written to DFHSHUNT during the tests.

The data presented in each sample begins with the SMF Type 88 data followed by a combination (hand crafted to reduce the presentation space) of data from RMF and the CICS 110 records. The RMF information is taken from the workload activity reports for the CICS and System Logger address spaces. The CICS-specific information is taken from summarized CICS 110 records using the CICS Performance Analyzer.

The sample on Figure 5-28 shows and explain the set od data collected for each run. These collection of data summarizes what is required to understand the CICS and system logger environment and should represent what kind of data you are collecting in your installation whenever you are evaluating this environment.

Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
DASD	0.4	8.3	0.2	.197	0.005	50.1582	50.0519	.0988	0.1	0	2.3	0.1	2571.42	1025.06	1546.36

Figure 5-28 Sample of the hand crafted reports



Using the example in Figure 5-28 on page 198, the data on the left is for the CICS address space. The TCB, DASD I/O, and APPL% are taken from the RMF report class data.

The number of transactions per second is taken from the transaction report class.

Average CPU per task, Average response time, Average JC Wait, and Average File Wait times are taken from the CICS PA reports.

The data on the right starting with LOGR TCB is for the System Logger address space (IXGLOGR), taken from the RMF report class reports.

The meaning of each heading is:

<b>Type</b>	The log stream type
<b>DASD</b>	Indicates this is a DASD-only log stream
<b>CF64</b>	Indicates this is for a CF-Structure log stream defined with MAXBUFSIZE(64000)
<b>CF32</b>	Indicates this is for a CF-Structure log stream defined with MAXBUFSIZE(32000)
<b>CICS TCB</b>	Reports the CICS TCB time in seconds for the interval, taken from the RMF report class information for the CICS address space.
<b>DASD I/O</b>	This field gives the average number of I/Os per second initiated by the region during the interval. The data is obtained from the RMF report class for the CICS region. Since the workload is based on VSAM files, this number provides a good indication of transaction throughput.
<b>APPL%</b>	Reports the percent of a CP used by the region in the report interval.
<b># Tran/Second</b>	The number of transactions per second is found in the RMF transaction report class and in the output of DFHOSTAT.
<b>Average CPU/Task</b>	The average CPU per transaction is taken from the CICS PA reports.
<b>Average Resp.</b>	The average transaction response time is provided in the CICS PA reports and the RMF transaction report class.
<b>Average JC Wait</b>	The average journal control wait time is found in the CICS PA reports and provides the average time tasks spent waiting for journal requests to complete.
<b>Average File WT</b>	This field reports the average time spent waiting for VSAM file requests to complete.
<b>LOGR TCB</b>	This field reports the TCB time consumed in the System Logger address space (IXGLOGR). The information is taken from the RMF report class for the System Logger address space.
<b>LOGR SRB</b>	This field provides the SRB CPU time used by the System Logger address space during the interval. The information is found in the RMF report class for the System Logger address space.
<b>DASD I/O</b>	This field gives the average number of I/Os per second initiated by the System Logger during the interval. The data is obtained from the RMF report class for the System Logger address space. This number is a good indicator of the amount of staging and offload data set activity.
<b>APPL%</b>	Reports the percent of a CP used by the System Logger address space in the report interval
<b>Storage Frames</b>	This field reports the average storage frames allocated to the System Logger address space during the interval. A storage frame is a 4K

page. The data is taken from the RMF report class for the System Logger.

**Idle Frames** The number of pages used when System Logger is idle, that is, prior to starting the CICS region(s).

**Net Frames** For System Logger, the increase due to CICS logging activity in the interval (represented by Net Frames) is calculated by subtracting the number of idle frames from the average storage frames.

**How Log Defer Interval (LGDFINT) can affect performance**

The LGDFINT CICS SIT parameter defines the log defer interval. The interval is the length of time, specified in milliseconds, the CICS log manager is to wait before invoking the System Logger for a forced write request. The value chosen may have an impact on transaction response time. Prior to CICS Transaction Server V2.2, the default is 30 milliseconds. With CICS Transaction Server V2.2, the default has been changed to 5 milliseconds.

To understand the effects of different LGDFINT values, three different runs of the workload has been executed with different set up using a DASD-only log stream, however the results are equally applicable to CF-Structure log streams. The following are the results of the three runs to illustrate the different behaviors. The log stream was defined with a HIGHOFFLOAD of 80, LOWOFFLOAD of 50 and a STG\_SIZE of 2518. AKPFREQ was set at 1000.

-LOGSTREAM NAME-	STRUCTURE NAME---	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES INVOKED	---# WRITES COMPLETED---			AVERAGE BUFFER SIZE						
		BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3							
-----EVENT-----															
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	OFF-	DASD	STRC	NTRY	STG	STG	RE-			
		INTERIM ST	W/O DASD	INTERIM ST	W/	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD			
		W/O DASD	WRITE	W/DASD	WRITE										
-----															
(1) AKPFREQ 1000 LGDFINT 30 STG_SIZE 2518 HIGHOFFLOAD 80 LOWOFFLOAD 50															
12/11/02 2:20:00 AM (SMF INTERVAL TIMESTAMP 'B8A8CC80A7400000'X)															
IYOT1.CICS22.DFHLOG *DASDONLY*															
		3032896	9834496	0	2401	0	0	0				1263			
		7585792	1852	0	0	1	0	0	0	1	0	0			
-----															
Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
DASD	1.6	37.8	0.6	1.57	.0044	11.11	11.042	0.0608	0.1	0.1	8.4	0.1	2949.35	859.09	2090.26
Logstream Writes/sec: 10.42															

Figure 5-29 Log Defer Sample 1

The first sample is shown in Figure 5-29. LGDFINT was set at 30 milliseconds.

At first glance, the SMF Type 88 data indicates the log stream is working fine; no data is being offloaded to the offload data set (BYT DELETD INTERIM ST W/DASD is zero). There was a single staging threshold and one offload.

The problem is not the log stream function, but rather the fact that very little work is being processed in the CICS region and the average response time is high.

From the RMF data, observe the CICS TCB time is very low (1.6 seconds), with a response time of 11.11 seconds. Also note CICS is only issuing an average of 37.8 DASD I/O per second. This DASD I/O has nothing to do with System Logger, but rather the CICS VSAM file accesses which is a direct reflection of the number of file control requests in the region. Finally notice the CICS region is only using 0.6% of an engine during this interval.

In the System Logger address space the CPU, SRB, and APPL% values are very low indicating a minimum of work is being passed to System Logger. The average number of DASD I/O is 8.4 per second, and the average net storage frames is 2090.26. The number of storage frames is an important value to watch, especially in an LPAR which is storage constrained. In this sample, the System Logger storage is 8.56M (2090.26 \* 4096) due to the CICS log stream.

The average response time per transaction is 11.11 seconds with an average CPU per task of 0.0044 seconds. Note, of the 11.11 seconds response time, an average of 11.042 seconds is spent waiting for journal I/O to complete, although there are no constraints in system logger.

The delay might be because CICS is waiting for the LGDFINT interval to expire before issuing and IXGWRITE. In the next run, LGDFINT will be decreased.

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN			#WRITES INVOKED	---# WRITES COMPLETED---			AVERAGE BUFFER SIZE						
		BY USERS	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3							
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----									
		INTERIM ST W/O DASD	W/O DASD WRITE	INTERIM ST W/DASD	W/ WRITE	OFF-LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	STG FULL	RE-BLD			
-----															
(2) AKPFREQ 1000 LGDFINT 5 STG_SIZE 2518 HIGHOFFLOAD 80 LOWOFFLOAD 50															
12/11/02 3:05:00 AM (SMF INTERVAL TIMESTAMP 'B8A8D68F92F00000'X)															
IYOT1.CICS22.DFHLOG *DASDONLY*		11240854	32935936	0	8041	0	0	0	0	0	1397				
		30801920	7520	1368064	334	4	0	0	0	4	0	0			
-----															
Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
DASD	5.4	140.4	2.2	4	.0039	3.0288	2.9631	0.059	0.1	0.2	27.3	0.1	2850.76	859.08	1991.68
Logstream Writes/sec: 29.85															

Figure 5-30 Log Defer sample 2

In the second sample, shown in Figure 5-30, LGDFINT was reduced to 5, all other parms remained the same. This change brought the transaction throughput up from 1.57 per second in sample 1 to 4 per second in sample 2, and the average response time has dropped from 11.11 to 3.0288 seconds.

When evaluating CICS transaction performance issues it's important to remember there are multiple parts in transaction response time. The two major categories are execution and suspend time. Execution time is when the application, CICS management modules, exit programs, and so on, are processing as part of the transaction. In the workload used, the suspend time is comprised of waiting for VSAM file I/O and log/journal writes. Wait for redispatch is the portion of suspend time after the ECB is posted until the task is redispatched.

Reducing the value for LGDFINT from 30 to 5 means each force write will only be held for 5 ms, rather than 30 before the IXGWRITE is issued. This reduces the opportunity for additional records to be added to the buffer, but it also reduces the time a task must wait for the log write to complete.

However, this region is capable of much better throughput. As outlined earlier, the workload consists of a very small assembler application which will generate 49 log writes per task (one for each read/update and one for each sync point). The data indicates journal wait time is still the largest component of the response time, accounting for 2.9631 seconds of the 3.0288 second response time.

-LOGSTREAM NAME-	STRUCTURE NAME---	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES INVOKED	---# WRITES COMPLETED----			AVERAGE BUFFER SIZE						
		BY USERS IXGWRITES	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3							
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----									
		INTERIM ST W/O DASD	W/O DASD WRITE	INTERIM ST W/DASD	W/ WRITE	OFF- LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	STG FULL	RE- BLD			
-----															
(3) AKPFREQ 1000 LGDFINT 00 STG_SIZE 2518 HIGHOFFLOAD 80 LOWOFFLOAD 50															
12/11/02 3:50:00 AM (SMF INTERVAL TIMESTAMP 'B8A8E09E7EA00000'X) )															
IYOT1.CICS22.DFHLOG *DASDONLY*		184275302	699846656	0	170861	0	0	0	0	0	1078				
		701902848	171363	0	0	87	0	0	0	156	0	0			
-----															
Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC	Wait	File	WT	TCB	SRB	I/O	Frames	Frames
DASD	85.3	2277	35.2	56.83	.0038	0.1845	0.1224	0.0556	0.2	3.9	572.8	1.8	3092.09	859.08	2233.01
												Logstream Writes/sec: 569.75			

Figure 5-31 Log Defer sample 3

Due to the workload and the journal wait time in Figure 5-30 on page 201, the log defer interval was removed by setting it to 00. Figure 5-31 shows some interesting results.

Setting LGDFINT to 0, means CICS will issue the IXGWRITE immediately after the log data is appended to the buffer.

Notice the number of IXGWRITES has increased from 8041 in sample 2 to 170861 in sample 3, an increase of over 21 fold. It's also interesting to note there are still multiple records per buffer (the average buffer size is 1078), indicating tasks are adding data to the buffers in the time it takes for each IXGWRITE to complete.

With the effective reduction in IXGWRITE time, there's a significant increase in transaction throughput. The transaction rate has increased from 4 transactions per second in sample 2 to 56.83 per second in sample 3, a 14 fold increase.

The average response time dropped to 0.1845, with a huge increase in the DASD I/O for both CICS and System Logger. The increase in CICS TCB time from 5.4 to 85.3 CPU seconds reflects a large increase in the amount of work being processed in the CICS region. Note that 0.1224 seconds of 0.1845 total seconds average response time per transaction are spent waiting for log writes to complete.

The increase in DASD I/O in the System Logger address space clearly reflects the fact CICS is issuing 569.75 log writes per second. From the RMF data (although not shown in Figure 5-31) the DASD requests in the System Logger address space are averaging 1.3 milliseconds.

The average System Logger storage frames for the CICS log streams has grown from 8.56M in the first sample to 9.15M in the third sample. The 0.59M increase is a small cost for the increase in throughput, increasing from 4 transactions a second in sample 2 to 56.83 transactions a second in sample 3.

**Tip:** The recommendation is to start with a LGDFINT value of 5 milliseconds for all releases. A value of 00 may be valid in some cases based on the speed of the processor, for example on a z900, the overall CICS workload and the amount of data log for each transaction.

The log defer interval value (LGDFINT) may be changed using CEMT I SYS. Evaluation of the results is much easier if the changes coincide with an SMF interval.

### ***How Activity Keypoint Frequency (AKPFREQ) can affect performance***

The other CICS parameter which can affect how CICS works with System Logger is AKPFREQ (Activity Keypoint Frequency).

Although activity keypoint frequency was used in CICS/ESA V4.1, it has taken on a more significant role in CICS Transaction Server. In CICS/ESA V4.1, AKPFREQ was a measure of the physical I/O issued to the system journal (DFHJ01). In CICS Transaction Server, it is a measure of the number of appends to the log stream buffer for DFHLOG. In CICS/ESA V4.1, it was common practice to specify AKPFREQ to provide an activity keypoint every 15 minutes. In CICS Transaction Server, the value should be set to ensure an activity keypoint is taken more frequently.

It's important to remember tail trimming is initiated for DFHLOG and DFHSHUNT during the CICS activity keypoint process. The value specified for AKPFREQ defines the frequency an activity keypoint is taken, therefore the frequency of tail trimming, which has a major impact on the data residency time. Trimming the tail of the log removes records which are no longer required for backout and recovery, therefore making space available in interim storage.

With a larger AKPFREQ value, the length of time between activity keypoints increases, therefore additional data is retained in interim storage. Either data will be offloaded, costing I/O and overhead to the offload data sets, or the size of interim storage will need to be increased, leading to additional storage requirements in the System Logger data space.

In a very busy region, it is not uncommon to observe an activity keypoint being taken every minute.

When HIGHOFFLOAD is reached, System Logger executes the following tasks as part of the offload process:

- ▶ Physically delete any data which has been logically deleted because of the AKPFREQ activity — and this is the goal for the DFHLOG and DFGSHUNT log stream.
- ▶ If the previous task does not take the log stream below the LOWOFFLOAD point, data is physically offloaded to the offload data set until the LOWOFFLOAD point is reached.
- ▶ Remove data no longer required from the data space.

To understand the effects of AKPFREQ, three samples are reported, all using a CF-Structure log stream, although the results are equally applicable to DASD-only log streams. Figure 5-32 shows the first sample's result where AKPFREQ was set to 7500, LGDFINT 00 and the log stream was connected to a 10M structure with HIGHOFFLOAD set to 80 and LOWOFFLOAD set to 50.

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES	---# WRITES COMPLETED----			AVERAGE					
		BY USERS	TO INTERIM	TO DASD		TYPE1	TYPE2	TYPE3		BUFFER				
		IXGWRITES	STORAGE	INVOKED					SIZE					
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----								
		INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-		
		W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD		
(7) AKPFREQ 7500 LGDFINT 00 10M structure HIGHOFFLOAD 80 LOWOFFLOAD 50														
08/25/03 1:55:00 PM (SMF INTERVAL TIMESTAMP 'B9EC872C550000'X)														
IYOT1.CICS22.DFHLOG	LOG_JG2_10M	332803618	571351296	137814804	1169531	951494	174502	43536	284					
		278135862	1004842	<u>121753484</u>	401533	324	18	0	****	0	0	0		
Type	CICS	DASD	APPL% # Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O	Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
CF64	179.0	<u>3550</u>	65.04	<u>65.04</u>	<u>.0063</u>	<u>0.1183</u>	<u>0.1032</u>	<u>0.5</u>	<u>12.2</u>	<u>101.1</u>	6.0	<u>14906.1</u>	<u>8933</u>	<u>5973.1</u>
Logstream Writes/sec: 2981.83														

Figure 5-32 Example 4 - AKPFREQ set to 7500

Examination of the SMF Type 88 data reveals 121753484 bytes were deleted after being moved to the offload data set. The number of NTRY FULL conditions was so large it does not fit in the report -- symbolized by the \*\*\*\*. Both conditions suggest either the data residency time is too long or the log stream size is too small. If this log stream were a DASD-only, the symptom would probably be STG FULL.

For this reason, in the next sample, the following changes have been made:

- ▶ CF structure storage size increased to 20MB
- ▶ AKPFREQ value reduced to 4000

The results are shown in Figure 5-33.

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES	---# WRITES COMPLETED----			AVERAGE						
		BY USERS	TO INTERIM	TO DASD		TYPE1	TYPE2	TYPE3		BUFFER					
		IXGWRITES	STORAGE	INVOKED					SIZE						
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----									
		INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-			
		W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD			
(8) AKPFREQ 4000 LGDFINT 00 20M structure HIGHOFFLOAD 80 LOWOFFLOAD 50															
08/25/03 2:05:00 PM (SMF INTERVAL TIMESTAMP 'B9EC896889B00000'X)															
IYOT1.CICS22.DFHLOG	LOG_JG2_10M	330844117	567308032	0	1159263	1147390	11873	0	285						
		330323643	1157378	0	0	302	0	0	0	0	0	0			
Type	CICS	DASD	APPL% # Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net	
	TCB	I/O	Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames	
CF64	173.7	<u>3529</u>	66.1	<u>73</u>	<u>0.0053</u>	<u>0.1191</u>	<u>0.0069</u>	<u>0.1035</u>	<u>0.2</u>	<u>15.3</u>	<u>90.1</u>	5.3	<u>15286</u>	<u>8933</u>	<u>6353</u>
Logstream Writes/sec: 3370.32															

Figure 5-33 Example 5 - AKPFREQ set to 4000

Examination of the SMF Type 88 data for this sample shows data is no longer being deleted after having been moved to the offload data set, with no negative events. The number of initiated offload processes has decreased to 302, indicating better use of interim storage. This

is the result of trimming the tail of the log stream more often, therefore making the offload process more efficient, that is, reaching the LOWOFFLOAD point without the need to move data to the offload datasets. Having 0 in the column 'BYT DELETED INTERIM ST W/DASD' means data which has been logically deleted by CICS was physically deleted, and the LOWOFFLOAD threshold was reached without further need of offloading data from the interim storage.

The System Logger TCB, SRB and APPL% remained constant for the period.

In the following sample, the AKPFREQ has been further reduced to a value of 2500 and you can observe the results in Figure 5-34.

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES INVOKED	---# WRITES COMPLETED---			AVERAGE BUFFER SIZE					
		BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3						
		BYT DELETED INTERIM ST W/O DASD	# DELETED W/O DASD WRITE	BYT DELETED INTERIM ST W/DASD	# DELETED W/ WRITE	-----EVENT-----								
						OFF- LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	STG FULL	RE- BLD		
(9) AKPFREQ <u>2500</u> LGDFINT 00 20M structure HIGHOFFLOAD 80 LOWOFFLOAD 50														
08/25/03 2:15:00 PM (SMF INTERVAL TIMESTAMP 'B9EC8BA4BE100000'X)														
IYOT1.CICS22.DFHLOG LOG_JG2_5M		332796138	570890752	0	1167419	1155716	11702	0	0	0	0	285		
		332663638	1166945	0	0	243	0	0	0	0	0	0		
Type	CICS	DASD	APPL% # Tran/ TCB I/O Second	Average CPU/Task	Average Resp	Average JC Wait	Average File WT	Logr TCB	Logr SRB	DASD I/O	APPL% Frames	Storage Frames	Idle Frames	Net Frames
CF64	178.9	<u>3545</u>	66.2 <u>77.38</u>	<u>0.0053</u>	<u>0.1185</u>	0.0069	0.1030	0.2	15.2	<u>92.1</u>	5.2	14844.5	8933	<u>5911.5</u>
Logstream Writes/sec: 3524.03														

Figure 5-34 Example 6 - AKPFREQ set to 2500

The SMF Type 88 data show the results of trimming the tail even more frequently. The log stream continues to run very well. The number of offloads is even lower because in this case system logger probably was capable to go beyond the lowoffload threshold boundary while physically deleting the records that have been logically deleted during the more frequent AKPFREQ activity. As consequence, also the number of TYPE 2 writes is down, and there are no negative events recorded.

The System Logger TCB, SRB and APPL% remained constant for the period.

The offload value in the SMF 88 records is the number of times the offload process is initiated. Notice, there are no negative event indicators; all offloads are due to reaching the HIGHOFFLOAD point. Good offloads are those initiated because we reach the HIGHOFFLOAD value. Bad offloads are the ones which are the result of things like structure full, and so on.

The negative side of reducing the AKPFREQ value is it increases the number of CSKP transactions (activity keypoint) which run in the interval.

There is a balance point between the number of activity key points taken, the log defer interval and the size of the log stream interim storage. Each should be considered when evaluating log stream performance.

Not every region responds in this manner, so each region should be evaluated individually. The number of activity keypoints should also be monitored using the DFHRM0205 messages. The intent is not to cause an excessive number of CSKP tasks to run, but to optimize the log stream operation. In a busy region, a target of one activity keypoint every 40 to 60 seconds is good.

Also, each DFHRM0205 message should be accompanied by a DFHLG0743 message indicating the log is being trimmed. If log tail trimming is not happening consistently, the transactions active during the interval should be investigated to understand why sync points are not being issued.

**Tip:** When changing the AKPFREQ (activity keypoint frequency) value using CEMT I SYS, it is best to make the changes coincide with an SMF interval, making it easier to evaluate the results of the change.

### Non-System Log Streams

User journals, forward recovery logs, and auto journals present a slightly different performance tuning opportunity. The data is always offloaded so the main considerations are interim storage residency time and how quickly the data can be offloaded.

The next samples show the effects of having a forward recovery log for one VSAM file in the region.

In sample 17 (Figure 5-35) there is a single CICS region, with DFHLOG allocated in a CF structure (LOG\_JG2\_20M). The region also has a forward recovery log (&applid..&SYSID..DFHJ03) for one of the files (FILEC). In the sample reported in Figure 5-35 the user journal was defined as a DASD-only log stream with STG\_SIZE of 25000 4K CIs, which can hold 102.4M of data.

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES	---# WRITES COMPLETED---			AVERAGE						
		BY USERS	TO INTERIM	TO DASD		TYPE1	TYPE2	TYPE3		BUFFER					
		IXGWITES	STORAGE		INVOKED				SIZE						
		BYT DELETD	# DELETES	BYT DELETD	# DELETS	-----EVENT-----									
		INTERIM ST	W/O DASD	INTERIM ST	W/	OFF-	DASD	STRC	NTRY	STG	STG	RE-			
		W/O DASD	WRITE	W/DASD	WRITE	LOAD	SHFT	FULL	FULL	THLD	FULL	BLD			
-----															
(17)	AKPFREQ 1000	LGDFINT 0	20M	HIGHOFFLOAD 80	LOWOFFLOAD 50	--- Forward Recovery Log			STG_SIZE 25000						
06/03/03 9:45:00 PM (SMF INTERVAL TIMESTAMP 'B984956CB0F00000'X)															
IYOT1.CICS22.DFHLOG	LOG_JG2_20M	418554024	720508928	0	1480321	1475486	4836	0	282						
		421389900	1490611	0	0	90	0	0	0	0	0	0			
IYOT1.IY01.DFHJ03	*DASDONLY*	30473250	101478400	29913580	24775	0	0	0	1230						
		<u>0</u>	<u>0</u>	<u>88190976</u>	<u>21531</u>	<u>2</u>	0	0	0	<u>268</u>	0	0			
Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
CF64	<u>215.3</u>	<u>4433</u>	79.7	<u>105.53</u>	.0062	.1007	.0060	.0842	0.1	17.6	83.4	6.0	31757.0	12760	<u>18997</u>
3619.24 logwrites/second															

Figure 5-35 User Journal - STG\_SIZE 25000, single region

System Logger is issuing DASD I/O at the rate of 83.4 per second. DFHLOG is defined as a CF log stream. No data is being offloaded from DFHLOG and it does not have a staging data set. Therefore in this case, all DASD I/O is associated with the user journal.

The greatest impact to the system is the 18997 page frames (77M) allocated by System Logger to contain the data for both CICS log streams. To be technically accurate, there are three log streams, however DFHSHUNT is connected but no data is being written. Notice the excellent transaction rate (105.53 per second) with a response time of 0.1007 seconds.

DFHLOG is running very well, and in fact DFHJ03 is also working very well but you can see some differences between the log streams. For the user journal, since all data is offloaded,



there should not be any information in the BYT DELETED INTERIM ST W/O DASD column. Likewise there should not be any data reported under #DELETES W/O DASD WRITE.

Under BYT DELETED INTERIM ST W/DASD will be the amount of data offloaded in the interval. Notice the number of STG THLD (the number of writes when the log stream was at or above the HIGHOFFLOAD threshold point) is over 130 times the number of offloads.

Because the log stream is so large CICS is able to issue 134 write requests in the time it takes System Logger to perform the offload. Using the average buffer size shown (1230), means CICS is writing 164820 bytes while the offload process is taking place.

To determine if that's a problem we must determine how much space is available above the HIGHOFFLOAD threshold. STG\_SIZE was specified as 25000 4K CIs, so using a HIGHOFFLOAD of 80%, the threshold point is at 20000 CIs. The storage above the threshold point is 5000 CIs or 20.48M. Since this is a DASD-only log stream and the average buffer size is less than the 4K, there is room for 5000 writes before filling the staging dataset. Remember each buffer is rounded to a 4K CI boundary.

In the sample reported in Figure 5-36 the sizing formula discussed under "Forward Recovery Logs" on page 184 was used to calculate the STG\_SIZE for the forward recovery log as 2549 (10.4M).

-LOGSTREAM NAME-	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES INVOKED	---# WRITES COMPLETED---			AVERAGE BUFFER SIZE						
		BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3							
		BYT DELETED INTERIM ST W/O DASD	# DELETES W/O DASD WRITE	BYT DELETED INTERIM ST W/DASD	# DELETES W/ WRITE	OFF- LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	STG FULL	RE- BLD			
-----															
(18) AKPFREQ 1000 LGDFINT 0 20M HIGHOFFLOAD 80 LOWOFFLOAD 50 --- Forward Recovery Log STG_SIZE 2549															
06/03/03 10:20:00 PM (SMF INTERVAL TIMESTAMP 'B9849D3F68400000'X)															
IYOT1.CICS22.DFHLOG	LOG_JG2_20M	445609444	767130624	0	1576179	1571029	5150	0	0	0	0	282			
		444776164	1573177	0	0	95	0	0	0	0	0	0			
IYOT1.IY01.DFHJ03	*DASDONLY*	29163300	97116160	31417260	23710	0	0	0	0	0	290	1230			
		0	0	93859840	22915	10	0	0	0	0	0	0			
Type	CICS	DASD	APPL%	# Tran/	Average	Average	Average	Average	Logr	Logr	DASD	APPL%	Storage	Idle	Net
	TCB	I/O		Second	CPU/Task	Resp	JC Wait	File WT	TCB	SRB	I/O		Frames	Frames	Frames
CF64	225.8	4720	83.3	112.36	.0061	.0921	.0061	.0751	0.1	19.1	79.9	6.5	22579.9	12044.2	10535.7
4533.17 logwrites/second															

Figure 5-36 User Journal - STG\_SIZE 2549, single region

Analysis of the data shows an increase in the offload number with an offload happening every 30 seconds, rather than every minute.

Compare the sample in Figure 5-36 with the sample in Figure 5-35 on page 206. The staging data set allocation decreased and consequently the number of frames required by System Logger (10535.7 versus 18997) to back the data in the data space has been decreased. Also, since the staging data set is smaller, we are reaching the HIGHOFFLOAD threshold more often, therefore causing the additional offload.

Figure 5-35 on page 206 sample illustrates a common error, where the size of a user journal is defined to contain the same amount of data as a journal in a R410 system. The nature of user journals is they are never accessed by the online CICS region, so all data will be offloaded, that is, logtail management does not apply to user journals. As such there is little

benefit in keeping the data in the interim storage. It is better to reduce the size (STG\_SIZE) so offloads happen on a regular basis, and let the data reside in the offload data sets.

LS\_SIZE should be specified large enough to contain the offloaded data during the prime processing period of the day.

Conclusion: STG\_SIZE for user journals should be calculated and specified to provide a data residency time of one minute or less, therefore reducing the amount of storage required by System Logger to back the data in the data space.

## **Summary**

Tuning CICS log streams for performance involves understanding the CICS region, the operating environment and workload. External factors, such as DASD response time and CF access time, make the task more challenging, but not insurmountable.

Data residency time is key to controlling the amount of storage required by the System Logger for duplexing the data to the data space. Data residency time is also key to establishing an even movement of data from non-system logs to the offload data sets.

Understand the tools which are available and how they can be used for log stream analysis.

As the environment changes, such as an increase in workload, a new level of the Coupling Facility Control Code, new applications, a new release of the operating system and a new release of CICS, ensure the log streams sizes are still valid.



## Other Logger exploiters

This chapter provides information about how the following exploiters use Logger:

- ▶ APPC Protected Conversations
- ▶ Merged Logrec
- ▶ OPERLOG
- ▶ Resource Recovery Services (RRS)
- ▶ System Automation for OS/390
- ▶ WebSphere Application Server

## 6.1 APPC protected conversations

APPC/MVS and Resource Recovery Services/MVS (RRS/MVS) introduce a new capability by providing system resource recovery services for APPC/MVS conversations. Some knowledge of resource recovery and its role is recommended to fully appreciate the value of APPC/MVS protected conversations. APPC/MVS protected conversations implements a communication resource manager on the z/OS platform and therefore enables z/OS to be a host for APPC conversations using the two-phase commit protocols.

From a cooperative application program's point of view, APPC/MVS protected conversations provide the capability for a cooperative application to have the updates in the cooperative applications coordinated. For example, each transaction program (TP) in the cooperative application can interact with database services and then request a coordinated commit or backout of resources that have been changed since the last point of consistency. RRS/MVS provides the ability for a cooperative application to invoke the commit and backout callable services. The role of APPC/MVS protected conversations in the two-phase commit protocol used to coordinate the update of local and remote resources, is to communicate the progress of the two-phase commit process to the distributed resource managers.

### 6.1.1 Functional description

You request APPC/MVS protected conversations through a VTAM® APPL definition statement for the APPC LU. The SYNCLVL keyword SYNCPT value must be specified for protected conversations. When you enable an APPC protected conversation, you must also define to System Logger a log stream for APPC/MVS use. APPC/MVS writes the results of a log name exchange with a partner LU into this log stream.

In fact, to provide resource recovery for protected conversations, APPC/MVS needs to know the names of local and partner LU log streams, and the negotiated sync point capabilities for each local/partner LU pair. This information needs to be available and accurate for successful resynchronization after a failure. To store this information, APPC/MVS uses a log stream.

APPC/MVS does not store conversation-specific information in the log stream. The log stream is used to store data related to unique partner LUs that have protected conversations with APPC/MVS. The log stream contains names of local and partner LU logs, and the negotiated sync point capabilities for each local/partner LU pair.

For each protected conversation, information is stored in RRS log streams during the two-phase commit process, but not in APPC/MVS log streams.

APPC/MVS can use both CF-Structure and DASD-only log streams. If you have workload using APPC/MVS protected conversation on multiple images within a sysplex, or if you plan to restart a workload using protected conversations on a different image, then you need to plan for a shared log stream using the CF as the interim media. If you are planning to use APPC/MVS from only one system, only one APPC/MVS will connect to the log stream, in which case you can use a DASD-only log stream if you wish. In this case, only APPC/MVS from this one system processes protected conversations. Other systems in a Parallel Sysplex can use APPC/MVS but they will fail to process any protected conversations since they will not be capable to connect to the log stream and will issue message ATB203I to document the return and reason codes received from the system logger IXGCONN service.

The APPC/MVS log stream is an active log stream and on an interval basis, APPC/MVS trims unneeded data from the log stream. There is no need to manage this log stream data using the System Logger parameters RETPD or AUTODELETE, or through any utility. It is

recommended that this log stream is defined with no RETPD and AUTODELETE keyword to avoid the deletion of data still in use by APPC/MVS.

There is one IXGWRITE in the log stream for each unique partner LU that uses protected conversations. So, for example, if you have 50 partner LUs, of which 25 have established protected conversations, you would have at least 25 IXGWrites. As you can see, there is not very frequent activity on this log stream. When an LUDEL is performed, the element is removed from the APPC log stream. When this is done, APPC invokes the purge log name affinity processing, which communicates with all partner LUs that have established protected conversations with this LU in the past. If the partner LU gives the OK to purge the affinities, then APPC deletes an entry in the log stream corresponding to that protected partner LU and its corresponding local LU. So, depending on the number of protected partner LUs and the size of the structure, this will determine where the data resides (CF or on DASD).

### Criticality/persistence of data

APPC/MVS keeps information about the conversation partners and their sync point in the log stream in order to be able to rebuild the conversation in case of a failure.

For this reason, APPC/MVS data is critical to the installation, and recovery from a loss of the data can be a complex process. To avoid a potential loss of data, we recommend that you use unconditional duplexing with this log stream. Duplexing the log stream should not cause a performance impact to the application since APPC/MVS doesn't update the log stream very frequently.

Losing data in the APPC/MVS log stream means that APPC/MVS loses all knowledge of all partners' log information and sync point capabilities. Refer to 6.1.4, "Recovery" on page 215, for more information about the impact of losing the APPC/MVS log stream data and the steps that need to be followed to restart the application.

### Log stream sizing

To size the interim media for the APPC/MVS log stream, you can use the following formula:

1. For each local LU that supports SYNCLVL=SYNCPT, calculate the maximum anticipated number of partner LUs that will communicate using protected conversations.
2. After identifying the number of partners for each local LU, add all the values together.
3. Multiple the resulting number by 300 bytes.

This is the *minimum* storage that should be allocated for the interim media for the APPC/MVS log stream. Refer to Example 6-1 for an example calculation.

#### *Example 6-1 APPC/MVS storage calculation sample*

---

There are 8 local LUs on system #0\$1, 3 of which are defined with SYNCLVL=SYNCPT (defined in VTAMLST)

- SYNPTLU1 communicates with up to 15000 partner LUs, but only 20% of those partners communicate using protected conversations.
- SYNPTLU2 communicates with up to 5000 partner LUs and all of them could use protected conversations.
- SYNPTLU3 communicates with 2000 partner LUs, of which about 50% use protected conversations.

Based on the above information, the minimum size for the interim storage for the APPC/MVS log stream is:  
 $(3000+5000+1000) \times 300 \text{ bytes} = 2,700,000 \text{ bytes}.$

---

## 6.1.2 Definition

### Subsystem definitions

There are no definitions to be done within APPC/MVS. If any protected conversations are started, APPC connects to a fixed name log stream: ATBAPPC.LU.LOGNAMES.

### APPC/MVS log stream structure definition in the CFRM Policy

If your installation decided to use coupling facility log streams for APPC/MVS, you need to make sure that the corresponding CF structure is defined in the CFRM policy and that the CFRM policy is activated in the sysplex through the SETXCF command. Example 6-2 contains a sample to define the CF structures for the APPC/MVS log stream. You should set SIZE to be roughly twice the size you determined in “Log stream sizing” on page 211.

This step is not required as you are using DASD-only log stream.

*Example 6-2 Sample of structure definition for APPC/MVS log stream*

---

```
//APPCSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(APPC_LOG)
  SIZE(8192)
  INITSIZE(4096)
  PREFLIST(FACIL01,FACIL02)
```

---

### Logger definitions

Even though you can use DASD-only log streams, most installations configure the APPC/MVS log stream as a CF-Structure log stream. For this reason, the following samples use a CF-Structure configuration. Refer to “APPC/MVS log stream structure definition in the CFRM Policy” on page 212 for more information about the CFRM parameters for the APPC/MVS log stream structure.

*Example 6-3 APPC/MVS log stream definitions*

---

```
//DEFINE EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE (LOGR)
  DEFINE STRUCTURE NAME(APPC_LOG)
    LOGSNUM(1)
    MAXBUFSIZE(65276)
    AVGBUFSIZE(248)

  DEFINE LOGSTREAM NAME(ATBAPPC.LU.LOGNAMES)
    STRUCTNAME(APPC_LOG)
    HIGHOFFLOAD(80)
    LOWOFFLOAD(40)
    DUPLEXMODE(UNCOND)
    STG_DUPLEX(YES)
    LS_DATACLAS(LOGR24K)
    STG_DATACLAS(LOGR4K)
    LS_SIZE(1024)
    HLQ(IXGLOGR)
    RETPD(0)
```

**LOGSNUM:** The LOGSNUM parameter controls the maximum number of log streams that may be allocated in the associated structure. We recommend specifying a LOGSNUM of 1, meaning that only one log stream will be placed in the structure.

**MAXBUFSIZE:** APPC/MVS requires a buffer size of 65276 bytes. If you use a MAXBUFSIZE value that is less than 65276, APPC/MVS issues message ATB209I and does not allow APPC/MVS LUs to handle any protected conversations until the buffer size is corrected and the LUs restarted.

**AVGBUFSIZE:** The APPC/MVS log stream contains one log block for each of the following:

- ▶ Each local/partner LU pair that has established a protected conversation.
- ▶ Each LU pair, if any, that has outstanding resynchronization work.

All log blocks are the same size: 248 bytes. Use 248 as the value for the average size of APPC/MVS log blocks.

**HIGHOFFLOAD:** The HIGHOFFLOAD value specifies how full the log stream interim storage should get before an offload process is initiated. We recommend using value not higher than 80%.

**LOWOFFLOAD:** The LOWOFFLOAD value defines the amount of data that is to be retained in the log stream interim storage following an offload process. In the APPC/MVS environment, the LOWOFFLOAD value should be between 40 to 60%.

**STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND):** If you suffer a loss of data in the APPC/MVS log stream, manual intervention is required to resolve the situation. To avoid this, we recommend defining the APPC/MVS log stream with STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND). Because APPC/MVS writes infrequently to its log, the performance impact should be minimal.

**STG\_DATACLAS:** Make sure you use an SMS data class that has share options of 3,3. Using share options other than this can result in failures when System Logger tries to use the staging data set.

You must use a data class that specifies a CI Size of 4 K.

**LS\_DATACLAS:** Make sure you use an SMS data class that has share options of 3,3. Using share options other than this can result in failures when System Logger tries to use the offload data set.

For optimal performance for the offload data sets, we recommend using a data class that specifies a CI Size of 24 K.

**STG\_SIZE:** The STG\_SIZE specifies the size, in 4K blocks, of the staging data sets for the log stream. You can omit this parameter and system logger will allocate a staging data set that will be equivalent to the coupling facility storage. In this case the staging data set is always in sync with the size of the coupling facility structure. Remember that having a staging data set smaller than the coupling facility structure might cause an offload even though the structure storage has not been totally used because the data set gets filled up earlier than the coupling facility storage.

**LS\_SIZE:** The LS\_SIZE specifies the size, in 4K blocks, of the offload data sets for the log stream. You can use an initial value of 1024 and then update depending on their usage.

**Attention:** Be sure you either specify an LS\_SIZE value on the log stream definition or you use an LS\_DATACLAS where you have space allocation defined. If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member or set via an ACS (Automatic Class Selection) routine. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Both methods are valid, it depends on your installation and on the variety of the log streams. If you do not have a big variety of offload data set, you can use SMS dataclas to manage the allocation space and simply omit the LS\_SIZE parameter. If your installation has multiple different type of offload data set that will generate too many SMS Dataclas, than you might want to consider to use LS\_SIZE on each log stream.

.HLQ: The HLQ specifies the high level qualifier for both the offload data sets and the staging data sets. If you do not specify a HLQ value, a default HLQ of IXGLOGR is used.

AUTODELETE and RETPD: It is very important that AUTODELETE(NO) and RETPD(0) are specified (or use the AUTODELETE and RETPD parameter defaults which are NO and 0, respectively) for the ATBAPPC.LU.LOGNAMES log stream. Using values other than these could result in data being deleted before APPC/MVS is finished with it, or in data being kept for far longer than necessary.

### Security definitions

The APPC/MVS log stream can be protected in the *Security Access Facility* (SAF) LOGSTRM class. The user ID associated with the APPC/MVS address space *must* have UPDATE access to the log stream. The RACF statements to define the profile and grant access to it are shown in Example 6-4.

*Example 6-4 Sample log stream security definitions*

---

```
RDEFINE LOGSTRM ATBAPPC.LU.LOGNAMES UACC(READ)
PERMIT ATBAPPC.LU.LOGNAMES CLASS(LOGSTRM) ID(APPC/MVS_userid) ACCESS(UPDATE)
SETROPTS CLASSACT(LOGSTRM)
```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

## 6.1.3 Operational considerations

The following is a consideration when operating in the APPC/MVS environment.

### Enq/Deq with OEM product

Verify that the PTF for APAR OA01793 is applied if you are using an OEM serialization product. In installations with OEM serialization products, where the qname list was not set up properly, the Global ENQ request made by APPC/MVS is treated as a Local ENQ, resulting in both APPC/MVSs making update request to System Logger and the last request being returned with an invalid version condition and message ATB2171 being issued with ABENDEC7.

This APAR will result in the ENQ on SYSZAPPC (APPC\_PPLU\_LOG) being treated as RNL(NO).



## Adding the APPC log stream after IPL time

If the LU is defined to accept protected conversations, via the SYNCLVL=SYNCPT and ATNLOSS=ALL in the VTAM APPL definitions, APPC/MVS attempts to connect to the APPC log stream at initialization of the LU (when LU is added for the first time, either via starting APPC or issuing the SET APPC=xx command). If the log stream does not exist, an operator message ATB208I is issued saying that the LU is active but not accepting protected conversations. The only remedy for this condition is to either issue an LUDEL and then an LUADD for the LU or start RRS if it was not started or, recycle RRS if it was already active.

The LU will not accept any protected conversations if the log stream is not defined. The program that attempted the protected conversation will get a Sync level not supported return code. DISPLAY APPC,LU,ALL will show SYNCPT=Y if the LU is syncpt-capable.

### 6.1.4 Recovery

You are not required to use staging data sets for APPC/MVS log data. However, if a system or CF failure causes the loss of APPC/MVS log data, warm/cold mismatches between local and partner LUs will result. To resolve such mismatches, you might have to:

1. Bring down APPC/MVS.
2. Use RRS ISPF panels to remove the expressions of interest that APPC/MVS LUs have in units of recovery.
3. Restart APPC/MVS.

This manual intervention might be more costly than the potential performance impact of using staging data sets. Because APPC/MVS writes infrequently to its log, the performance impact should be minimal, and the use of STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND) ensures that no data will be lost even if you lose both the z/OS system and the CF containing the APC/MVS log stream.

### 6.1.5 Performance

After your initial log streams allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments are necessary to your installation:

- ▶ SMF 88 data produced by the System Logger
- ▶ SMF 70-78 data produced by various z/OS components

There are several tools which aide in the analysis of log stream performance problems:

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70 -78 records

APPC/MVS does not have a lot of activity. For this reason performance are rarely an issue for this log stream.

The main fields in SMF88 that you should pay attention for this log stream are the OFFLOAD, DASD SHIFT, STRUCTURE FULL, ENTRY FULL and STAGING THRESHOLD. Refer to 8.6.3, "SMF Type 88 records and IXGRPT1 program" on page 291 for a description of these fields and what to do in case of one of this condition happens in your installation.

## 6.2 Logrec

The Logrec log stream is an alternative to the use of LOGREC data sets. The LOGREC data sets normally contain an accumulation of records (called LOGREC records) describing errors that have occurred on the system. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions. The use of the log stream instead of LOGREC data sets allows you to have a single source of information for errors across all systems in the sysplex.

### 6.2.1 Functional description

You can choose where the system will record Logrec error records. When a system is not in a sysplex, an installation can use a LOGREC data set, associated with an individual system, to record error records.

In a sysplex, however, because each system requires its own Logrec data set, you might need to look at each LOGREC data set when an error occurs.

To eliminate the problem of having to manage multiple LOGREC data sets, an installation can choose to define one CF-Structure Logrec log stream. Using a CF-Structure Logrec log stream eliminates the following:

- ▶ Running IFCDIP00 to initialize multiple LOGREC data sets
- ▶ Having the LOGREC data set fill up on you
- ▶ Scheduling the daily offload of LOGREC data sets
- ▶ Concatenating multiple history data sets
- ▶ Archiving Logrec records

The Logrec log stream is a funnel-type log stream. Each system writes records as they get generated. The records never get referenced unless the installation browses the log stream to retrieve information about specific errors.

The name of the Logrec log stream is fixed: SYSPLEX.LOGREC.ALLRECS. There can only be one Logrec log stream in a sysplex, but each system can independently choose to record the error data in the log stream or in the LOGREC data set.

#### **Criticality/persistence of data**

While Logrec data is a valuable tool to help you diagnose problems, the data would not be considered critical. Loss of the data will not result in any adverse impact on the system or its availability.

Logrec keeps the log records in the log stream merged from all the systems in the sysplex that are connected to the log stream. In order to be able to fall back to the use of the LOGREC data sets, we recommend that you IPL with a LOGREC data set initialized by IFCDIP00 and then immediately switch to the Logrec log stream via a command. If you do not IPL with a Logrec data set, you will not be able to change the Logrec recording medium from LOGSTREAM to DATASET using the SETLOGRC command.

If you wish to share the Logrec across multiple z/OS instances, the log stream must be placed in a CF structure. It is usually mapped to a dedicated Coupling Facility structure. In a single system sysplex you can use a DASD-only log stream. This provides similar benefits to a CF log stream, with the exception of course that it only contains data from a single system.

Logrec data can reside in the log stream indefinitely. If you choose this option, you need to pay attention to the size of the offload data sets and on how to manage them. You need sufficiently large offload data sets that you do not generate so many that you run out of

DSEXTENTS (by default you get up to 168 offload data sets per log stream). In this case, you will probably need some means of migrating older data sets using a product like HSM. This automatic migration will provide some relief in space consumption if you are planning to keep the logrec records for a long time. However, if your installation keeps these records for a long time (a number of years, for example), you might want to consider whether the log stream is the correct archival place for the Logrec data. If not, you can use a utility to move data from the log stream to a GDG data set, for example.

### Log stream sizing

Logrec is a funnel type log stream; that is, it never deletes old records itself. Regardless of the size of the structure containing the log stream, eventually the data will be offloaded to an offload data set - how many there are and how frequently it happens will depend on the overall configuration of the Logrec environment.

For an initial sizing of your Logrec log stream, check how many records are written per second to the Logrec data set on each of your systems, aggregate them, and use the CFSizer tool to determine the size of the CF structure. The tool can be located at the IBM Web site:

<http://www.ibm.com/servers/eserver/zseries/ps>

## 6.2.2 Definition

### Subsystem definition

The suggested way to set up the Logrec environment is to IPL each system using its own Logrec data set specified in the IEASYS parmlib member as in Example 6-5.

*Example 6-5 IEASYS Parmlib member for Logrec data set*

---

```
LOGCLS=L,  
LOGLMT=010000,  
LOGREC=SYS1.&SYSNAME..LOGREC,  
MAXUSER=128,  
MLPA=00,
```

---

You can then switch to using the log stream by using the SETLOGRC command. This process allows your installation to fall back to using the data set should you so wish.

Alternately, you can choose to use the log stream as the recording medium immediately from the IPL. In this case, you specify LOGREC=LOGSTREAM in the IEASYS Parmlib member as shown in Example 6-6.

*Example 6-6 IEASYS Parmlib member for log stream*

---

```
LOGCLS=L,  
LOGLMT=010000,  
LOGREC=LOGSTREAM,  
MAXUSER=128,  
MLPA=00,
```

---

In this configuration, if you become unable to use the log stream for some reason, the system will lose the recording capability. This means that the system will continue to run, but any Logrec records that are generated will be lost. However, if the log stream is not available at IPL time, the system doesn't complete initialization.

## Logrec log stream structure definition in the CFRM Policy

Before using the Logrec log stream, you have to make sure that the corresponding CF structure has already been defined in the CFRM policy and that the CFRM policy has been activated in the sysplex through the SETXCF command. Example 6-7 contains a sample to define the Coupling Facility structures for the Logrec log stream.

### *Example 6-7 Structure definition for Logrec*

---

```
//LOGREC JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(LOGREC)
  SIZE(2048)
  INITSIZE(1024)
  PREFLIST(FACIL01,FACIL02)
```

---

## Logger definitions

Most installations use a CF-Structure log stream for Logrec. For this reason, the following is an example on how to define the Logrec log stream into the LOGR policy. Refer to “Logrec log stream structure definition in the CFRM Policy” on page 218 for the further informations on CFRM parameters for the Logrec structure.

### *Example 6-8 Logger policy definition sample for Logrec log stream*

---

```
//DEFINE EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE (LOGR)
  DEFINE STRUCTURE NAME(LOGREC)
    LOGSNUM(1)
    AVGBUFSIZE(4068)
    MAXBUFSIZE(4068)

  DEFINE LOGSTREAM NAME(SYSPLEX.LOGREC.ALLRECS)
    STRUCTNAME (LOGREC)
    LS_DATACLAS(LOGR4K)
    HLQ(IXGLOGR)
    LS_SIZE(1024)
    LOWOFFLOAD(0)
    HIGHOFFLOAD(80)
    STG_DUPLEX(NO)
    RETPD(0)
    AUTODELETE(NO)
```

---

**MAXBUFSIZE:** MAXBUFSIZE must be at least 4068 because Logrec writes records in one page blocks.

**AVGBUFSIZE:** On the structure definition, AVGBUFSIZE specifies the average size, in bytes, of individual log blocks that can be written to log streams allocated to the coupling facility. You can use the 4068 value suggested that fits most of the installation.

**LOWOFFLOAD:** The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the Logrec environment, the LOWOFFLOAD value should be 0 since there is no need to keep any data in the interim storage.

LS\_SIZE: The LS\_SIZE specifies the size, in 4 K blocks, of the log stream offload data sets for the log stream. An initial value for the LS\_SIZE is to allocate as much space as is allocated for all the Logrec data sets on the systems in the sysplex before migrating to a Logrec log stream.

**Caution:** Be sure you either specify an LS\_SIZE value on the log stream definition or you use an LS\_DATACLAS where you have space allocation defined. If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Both methods are valid, it depends on your installation and on the variety of the log streams. If you do not have a big variety of offload data set, you can use SMS dataclas to manage the allocation space and simply omit the LS\_SIZE parameter. If your installation has multiple different type of offload data set that will generate too many SMS Dataclas, than you might want to consider to use LS\_SIZE on each log stream

HLQ: The HLQ specifies the high level qualifier for both the log stream data set name (and the staging data set name if used). If you do not specify a high level qualifier, a high level qualifier IXGLOGR is defaulted.

AUTODELETE and RETPD: If your installation is planning to use the Logrec log stream as the accumulation place for the errors records, you should specify AUTODELETE(YES) and RETPD(x), where x is the longest time you would like to keep the data in your installation. In such a configuration, it is recommended that you migrate the offload data sets to avoid high usage of DASD space. At the beginning, when you activate this configuration, you should monitor the amount of offload data sets that are created to avoid potential directory full condition on the LOGR inventory couple data set.

### Security definitions

The Logrec log stream can be protected in the *Security Access Facility* (SAF) LOGSTRM class. You can provide default READ access so all the users will be able to browse and retrieve records from this log stream. If you have batch job to archive Logrec records, you need to give UPDATE permit to that user ID in order to perform deletion of records.

#### *Example 6-9 Sample log stream security definitions*

---

```
RDEFINE LOGSTRM SYSPLEX.LOGREC.ALLRECS UACC(READ)
SETROPTS CLASSACT(LOGSTRM)
```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

## 6.2.3 Operational considerations

The following are considerations on how to operate in the Logrec environment.

## Displaying Logrec status

The status of Logrec recording can be displayed using D Logrec command (Example 6-10).

### Example 6-10 D Logrec command

---

```
D Logrec
IFB090I 10.09.40 Logrec DISPLAY 276
CURRENT MEDIUM = LOGSTREAM
MEDIUM NAME = SYSPLEX.LOGREC.ALLRECS
STATUS = CONNECTED
```

---

Where CURRENT MEDIUM can be:

- ▶ **IGNORE** Recording of Logrec error and environmental records is disabled.
- ▶ **LOGSTREAM** The current medium for recording Logrec error and environmental records is a log stream.
- ▶ **DATASET** The current medium for recording Logrec error and environmental records is data set.

MEDIUM NAME can be:

- ▶ **SYSPLEX.LOGREC.ALLRECS** (if the current medium is LOGSTREAM)
- ▶ The data set name where Logrec errors are recorded (if the current medium is DATASET)

The STATUS line is only displayed if the current medium is LOGSTREAM. Its values can be:

- ▶ **CONNECTED**
- ▶ **NOT CONNECTED**
- ▶ **LOGGER DISABLED**

If the STATUS shows as **CONNECTED**, then the log stream is connected and active.

## Changing Logrec recording medium

Logrec recording can be changed dynamically via:

```
SETLOGRC {LOGSTREAM|DATASET|IGNORE}
```

Where the operands indicate:

- ▶ **LOGSTREAM**: The desired medium for recording Logrec error and environmental records is a log stream. To use a log stream in your installation, the Logrec log stream must be defined.
- ▶ **DATASET**: The desired medium for recording Logrec error and environmental records is a data set. Setting the medium to data set works only if the system had originally been IPLed with a data set as the Logrec recording medium. If the system was not IPLed with a data set Logrec recording medium and the attempt is made to change to DATASET, the system rejects the attempt and maintains the current Logrec recording medium.
- ▶ **IGNORE**: This indicates that recording Logrec error and environmental records is to be disabled. It is not recommended to use the IGNORE option except in a test environment.

## Running EREP

There is a sample job, IFBEREPS, in SYS1.SAMPLIB that can be used as a template for running EREP jobs. It must be modified to match your installation's setup. The job can be used to extract data from the Logrec log stream, create history data sets, and clear records from the log stream.

A sample job which creates a system exception report is shown in Example 6-11.

*Example 6-11 Sample EREP job to browse Logrec log stream*

---

```
// EREP JOB (999,POK),MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID,  
// CLASS=A,REGION=4M  
// EREPSER EXEC PGM=IFCEREPI,REGION=4M,PARM='CARD'  
// ACCIN DD DSN=SYSPLEX.Logrec.ALLRECS,  
// SUBSYS=(LOGR,IFBSEXIT,, 'DEVICESTATS'),  
// DCB=(RECFM=VB,BLKSIZE=4000)  
// ACCDEV DD DUMMY  
// SERLOG DD DUMMY  
// DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,15,,CONTIG)  
// TOURIST DD SYSOUT=*,DCB=BLKSIZE=133  
// EREPPT DD SYSOUT=*,DCB=BLKSIZE=133  
// SYSABEND DD SYSOUT=*  
// SYSIN DD *  
    HIST  
    ACC=N  
    SYSEXN
```

---

## 6.2.4 Recovery

As described in 6.2.1, “Functional description” on page 216, your installation can prevent any loss of data condition if the configuration is planned carefully. Follow the suggested method to implement and to activate the Logrec recording on the log stream media.

## 6.2.5 Performance

After your initial log streams allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments are necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70-78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70–78 records

Usually this is not a critical log stream from a performance perspective. There are some fields in the SMF88 that you might want to check against this log stream: DASD SHIFT, STRUCTURE FULL, ENTRY FULL. Refer to 8.6.3, “SMF Type 88 records and IXGRPT1 program” on page 291 for a description of these fields and what to do in case of one of this condition happens in your installation.

## 6.3 OPERLOG

The MVS operations log (OPERLOG) function of the MVS console services component provides a sysplex-wide merged message log by using the MVS system logger services. There is only one OPERLOG log stream within the entire sysplex but the recording of the messages into the log stream can be activated on a system basis. You might have systems that need to merge in the OPERLOG log stream and other systems that can continue to record independently on their syslog.

### 6.3.1 Functional description

The OPERLOG log stream is a funnel-type log stream where each system connected to this log stream puts its own messages.

The messages are logged in the form of message data blocks (MDB). An MDB is a structured control block that contains a complete representation of a message, including both text and control information. These messages are written into the log stream in chronological order as they are received and processed by system logger services; this does not assure that the message are written in the log stream in the order they are generated since they might get to the logger in a different order than the one they are generated. This might be visible when you retrieve records and you can see that records from different systems are not ordered correctly by time stamp. This process can also be influenced by the speed of the processor where the application using logger services is actually running.

#### Criticality/persistence of data

OPERLOG keeps the log records in the log stream merged from all the systems in the sysplex that are connected to the log stream. Since a system can record to both OPERLOG and SYSLOG and if a system cannot record to the OPERLOG log stream it automatically switches to SYSLOG even though SYSLOG was not activated, data on the OPERLOG log stream are not considered to be critical. Losing the OPERLOG capability forces the installation to fall back to individual syslog.

OPERLOG is anyway not considered the final archival place for operations log data. Usually, installations keep the active portion of the operations data in OPERLOG, that is, for a week, and then retrieve the data from the log stream to be archived on more traditional media and readable format.

OPERLOG is usually mapped to a dedicated Coupling Facility structure. You can control OPERLOG duration of the data:

- ▶ Through the log stream definitions AUTODELETE and RETPD if you want to let system logger perform the cleaning of the unwanted records
- ▶ Through your installation scheduling environment by using the IEAMDBLG Program. Refer to “IEAMDBLG Program” on page 228 for more details.

#### Log stream sizing

By being a funnel type log stream, OPERLOG relies on the offload process to accumulate data in time. So, no matter the size of the structure mapping the log stream, you should always expect offload at a certain point, how many and how frequent depending on the overall configuration of the OPERLOG environment.

For an initial sizing of your OPERLOG log stream, you can simply go in your current environment and see how many writes per second are happening currently on each of your systems (that is, using SDSF), aggregate them and use the Web-based tool to determine the size of the coupling facility structure. The tool can be located at the IBM Web site:

<http://www.ibm.com/servers/eserver/zseries/ps0>

Once that you enable the function, you can use the RMF Coupling Facility Report and SMF88 data to understand if the structure size and offload data sets size require any adjustments.

### 6.3.2 Definition

In this section we review definitions.



## Subsystem definition

You can record system messages and commands to the system log (SYSLOG), the operations log (OPERLOG), or an MCS printer. The recording of the messages and commands is controlled by the hardcopy message set. The hardcopy message set represents messages that can be recorded in hardcopy on these media. It is defined in the HARDCPY definition in Parmlib CONSOLxx member. OPERLOG is defined as part of the hardcopy setup.

```
HARDCOPY DEVNUM {(devnum) },
{(SYSLOG) },
{(OPERLOG) },
{(devnum,OPERLOG)},
{(SYSLOG,OPERLOG)}
```

To enable an image to use the operlog, you can defined the set up in the CONSOLxx parmlib member or use the VARY command to activate or deactivate it after IPL has completed.

The name of the OPERLOG log stream is fixed, SYSPLEX.OPERLOG. So, when a system is enabled to use the OPERLOG log stream, it will connect to SYSPLEX.OPERLOG log stream.

## Structure definition in the CFRM Policy for the OPERLOG log stream

Before using the OPERLOG log stream, you need to define that the correspondent Coupling Facility structure in the CFRM policy and make sure that the CFRM policy has been activated in the sysplex through the SETXCF command. Here is an initial sample to define the Coupling Facility structures for the OPERLOG log stream.

*Example 6-12 Structure definition for OPERLOG sample*

---

```
//OPERLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(OPERLOG)
  SIZE(40448)
  INITSIZE(40448)
  PREFLIST(FACIL01,FACIL02)
```

---

## Logger definition

Most of the installation use OPERLOG as a coupling facility log stream, although you can use a DASD-only log stream if you re running a monoplex or single system sysplex or if you want to enable OPERLOG only on one system in your sysplex. The following is an example of how to define the log stream to the LOGR policy. Remember, before executing the LOGR policy, check that the corresponding CFRM structure has been defined in the CFRM policy and activated in the sysplex. Refer to “Security definitions” on page 226 for the further informations on CFRM parameters for the OPERLOG structure.

*Example 6-13 OPERLOG log stream definitions for the LOGR policy sample*

---

```
//OPERLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR)
  DEFINE STRUCTURE NAME(OPERLOG)
  LOGSNUM(1)
  MAXBUFSIZE(4096)
```

---

```

AVGBUFSIZE(512)

DEFINE LOGSTREAM NAME(SYSPLEX.OPERLOG)
  STRUCTNAME(OPERLOG)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  LOWOFFLOAD(0)
  HIGHOFFLOAD(80)
  STG_DUPLEX(NO)
  RETPD(Note1)
  AUTODELETE(Note1)

```

---

**LOGSNUM:** On the structure definition, LOGSNUM specifies the number of log streams that you want to allocate to the same coupling facility structure. The value you specify determines how many pieces the structure is divided into and how much room there is in each piece. For OPERLOG, the suggested mapping is one log stream per structure.

**MAXBUFSIZE:** MAXBUFSIZE should be 4096 for OPERLOG records.

**AVGBUFSIZE:** On the structure definition, AVGBUFSIZE specifies the average size, in bytes, of individual log blocks (MDBs) that can be written to log streams allocated to the coupling facility. You can use the 512 value suggested that fits most of the installation or you can use the following method to evaluate the correct value for your installation.

To determine the MDB size in the operation log stream, extract data from the OPERLOG log stream using the LOGR subsystem data set interface. You can use the LOGR subsystem for existing eligible applications that need to access log stream data in data set format. The JCL for the MDB extract job are as shown in Example 6-14.

*Example 6-14 JCL to evaluate OPERLOG AVGBUFSIZE*

---

```

//RUN1 JOB MSGLEVEL=1,MSGCLASS=X,CLASS=A
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=X
//SYSUT2 DD DSN=out.data.set,DISP=SHR
//SYSUT1 DD DSN=SYSPLEX.OPERLOG,
// SUBSYS=(LOGR,IXGSEXIT,'FROM=OLDEST'),
// RECFM=VB,BLKSIZE=12292,LRECL=4096,DSORG=PS
//SYSIN DD DUMMY

```

---

From the IEBGENER output, the average MDB size was 338.21 bytes. The MDB size distribution was:

```

200> <=300> <=400> <=500> <=600> <=700> <=800> <=900> <=1000> <=1100> <1200> <=1300> <1400>
78998 65102 32308 1001 478 142 359 114 156 95 82 59 997
Maximum MDB size 60333 Minimum MDB size 234 Number of samples 179891

```

---

The IEBGENER utility program SYSUT1 DD-statement in the previous JCL allocates the OPERLOG log stream through the subsystem LOGR. You can use the above value to define your log stream in your LOGR policy.

**HIGHOFFLOAD:** The HIGHOFFLOAD parameter is used to determine when the space dedicated to the log stream in the Coupling Facility is filling up and an offload need to be initiated to re-gain available space. Since the OPERLOG log stream is a funnel type log stream, HIGHOFFLOAD should be set at 80% at least initially and then use the SMF88 report to evaluate if this value need same tuning.

**LOWOFFLOAD:** The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the OPERLOG environment, the LOWOFFLOAD value should be 0.

**LS\_DATACLAS:** With this keyword, you can specify the SMS Dataclass that is used for the allocation of offload data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure the desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the share options. Offload data sets may use a CI size up to 24K; in general, the larger CI size provides better performance, however, if the log stream consists mostly of log blocks of a much smaller size, the 24K CI size may be wasteful

**LS\_SIZE:** The LS\_SIZE specifies the size, in 4-K blocks, of the log stream DASD data sets for the log stream.

**Caution:** Be sure you either specify an LS\_SIZE value on the log stream definition or you use an LS\_DATACLAS where you have space allocation defined. If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Both methods are valid, it depends on your installation and on the variety of the log streams. If you do not have a big variety of offload data set, you can use SMS dataclas to manage the allocation space and simply omit the LS\_SIZE parameter. If your installation has multiple different type of offload data set that will generate too many SMS Dataclas, than you might want to consider to use LS\_SIZE on each log stream.

**HLQ:** The HLQ specifies the high level qualifier for both the offload data set name (and the staging data set name if used). If you do not specify a high level qualifier, a high level qualifier IXGLOGR is defaulted. The data set names format for the OPERLOG log stream data sets is hlq.SYSPLEX.OPERLOG.Annnnnnn where nnnnnnn is a running sequence number.

**STG\_DUPLEX and DUPLEXMODE:** DUPLEX and DUPLEXMODE determine under what conditions, if any, System Logger uses a staging data set for a given log stream on a given system.

**STG\_DUPLEX(NO)** indicates that System Logger never uses staging data sets, regardless of the nature of the connection to the structure on which the log stream resides.

**STG\_DUPLEX(YES)** with **DUPLEXMODE(UNCOND)** states that System Logger should always use staging data sets for the log stream, even if the connection to the associated structure is failure independent. Even though this guarantees that there will never be a loss of data condition on the log stream, it comes at a price: every write to the log stream is gated by the speed of the I/O to the staging data set.

**STG\_DUPLEX(YES)** with **DUPLEXMODE(COND)** means that System Logger only uses a staging data set if the CF on which the structure resides becomes volatile, or the structure resides in the same failure domain as the System Logger system. If none of these conditions exists, System Logger duplexes the log blocks in the System Logger data space. With this setting, you only pay the penalty of writing to a staging data set in those conditions that might reside in a loss of data condition due to a single failure.

Since OPERLOG can tolerate a loss of data condition, your installation can decide how sensitive is this data and set up the duplexing consequently.

AUTODELETE and RETPD: For the OPERLOG log stream, you can handle the expiration of the oldest records in different ways, mainly depending upon the need of archiving or not of the operations data.

**Note:** If you do not need to keep operations data, then the suggested way is to let system logger delete old records once they expire by specifying the AUTODELET(YES) and RETPD>0 parameters on the log stream definition.

If your installation needs to maintain operations data for a long time, then it is probably suggested to move the data out of the log stream, put then in a readable format and store them on a traditional support, that is, tape. In this case, you might specify AUTODELETE(NO) and RETPD(0) and then use the IEAMDBLG Program to archive the data on your installation schedule. Any time data are moved from the log stream to the new support, IEAMDBLG also deletes the old records since they are not needed any longer.

### Security definitions

The OPERLOG log stream can be protected in the *Security Access Facility* (SAF) LOGSTRM class. OPERLOG log stream should have READ access as default so everybody can browse the operlog data and UPDATE access level only to those users that need the capability to delete records from the log stream, that is, the user ID associated with the job running the IEAMDBLG program.

*Example 6-15 Sample log stream security definitions*

---

```
REDEFINE LOGSTRM SYSplex.OPERLOG UACC(READ)
PERMIT SYSplex.OPERLOG CLASS(LOGSTRM) ID(userid1) ACCESS(UPDATE)
SETROPTS CLASSACT(LOGSTRM)
```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

### 6.3.3 Operational considerations

The following are items to consider when you are using OPERLOG.

#### **Enabling/Disabling OPERLOG**

OPERLOG can be turned on and off on each system within the sysplex with the following command, providing the MVS system logger is activated (Example 6-16).

*Example 6-16 VARY command sample*

---

```
V OPERLOG,HARDCPY
  IEE889I 20.59.31 CONSOLE DISPLAY 153
  MSG: CURR=10 LIM=1500 RPLY:CURR=6 LIM=999 SYS=SC50 PFK=00
  CONSOLE/ALT ID ----- SPECIFICATIONS -----
  SYSLOG COND=H AUTH=CMDS NBUF=2 UD=Y
  ROUTCDE=ALL
  OPERLOG COND=H AUTH=CMDS NBUF=N/A UD=Y
  ROUTCDE=ALL

V OPERLOG,HARDCPY,OFF
  IEE338I OPERLOG INACTIVE AS HARDCPY
```

---

The intended scope of an OPERLOG is a sysplex. However, a given system in the sysplex may or may not be using OPERLOG at any particular time. The operator commands that control the status of OPERLOG and the initialization parameter that activates it at IPL have a

single system scope. Furthermore, a failure in OPERLOG processing on one system does not have any direct effect on the other systems. The result is that an OPERLOG may contain records from an entire sysplex or from only a subset of the systems, depending on the installation's requirements and on the environmental factors.

**SDSF considerations**

The following considerations apply when using SDSF and OPERLOG:

- ▶ You can understand if you are browsing OPERLOG or SYSLOG by looking on the upper left corner of the log display. SDSF displays which media is currently browsing and you can switch from OPERLOG to SYSLOG by using the LOG OPERLOG or LOG SYSLOG command at any time.
- ▶ If you are browsing the OPERLOG while structure rebuild takes place either due to a failure or to maintenance procedure, the log stream data will be unavailable for the time it takes to rebuild the log stream in the alternate coupling facility. For the duration of this interval, you are notified that data is not available with an error message in your upper right corner as shown in Figure 6-1.

```

Display Filter View Print Options Help
-----
SDSF OPERLOG DATE 04/28/2003 1 WTOR LOG BROWSE ERR 0008-0861
COMMAND INPUT ==> SCROLL ==> PAGE
000290 SETHCF START,RB,STRNM=LOG_TEST_001
000080 IXC521I REBUILD FOR STRUCTURE LOG_TEST_001 502
000080 HAS BEEN STARTED
000080 IXC367I THE SETHCF START REBUILD REQUEST FOR STRUCTURE 503
000080 LOG_TEST_001 WAS ACCEPTED.
001 REPLY WITH VALID MCF SYSTEM OPERATOR COMMAND
***** BOTTOM OF DATA *****
  
```

Figure 6-1 Log stream rebuild processing

- ▶ In order to deallocate the OPERLOG log stream, all the active connections should be removed. Connections to the OPERLOG log stream are mainly SDSF users and until the last TSO user exits from SDSF, the log stream cannot be deallocated. You can discover which user is currently connected to the log stream by using the DISPLAY LOGGER command as shown in Figure 6-2.

```

D LOGGER,C,LSNAME=SYSPLEX.OPERLOG,DETAIL
IXG601I 12.28.55 LOGGER DISPLAY 389
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #@$2
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----
SYSPLEX.OPERLOG    LOG_TEST_001        000005  IN USE
DUPLIXING: STRUCTURE, STAGING DATA SET
JOBNAME: SICA      ASID: 0029
R/W CONN: 000001 / 000000
RES MGR./CONNECTED: *NONE* / NO
IMPORT CONNECT: NO
JOBNAME: ANANIA   ASID: 003B
R/W CONN: 000002 / 000000
RES MGR./CONNECTED: *NONE* / NO
IMPORT CONNECT: NO
JOBNAME: KYNEF    ASID: 0022
R/W CONN: 000001 / 000000
RES MGR./CONNECTED: *NONE* / NO
IMPORT CONNECT: NO
JOBNAME: CONSOLE  ASID: 000A
R/W CONN: 000000 / 000001
RES MGR./CONNECTED: *NONE* / NO
IMPORT CONNECT: NO
  
```

Figure 6-2 Display Logger,Connection

### ***IEAMDBLG Program***

SYS1.SAMPLIB contains a sample program (IEAMDBLG) to read log blocks from the OPERLOG log stream and convert them to SYSLOG format. The program is an example of how to use the services of the MVS system logger to retrieve and delete records from the OPERLOG stream. It reads the records created in a given time span, converts them from Message Data Block (MDB) format to Hard-copy Log format (HCL or JES2 SYSLOG), and writes the SYSLOG-format records to a file. It also has an option to delete from the stream all the records created prior to a given date. When you use the delete option, a suggestion is to first copy the records on an alternate media and then conditionally delete them on a separate JCL step: this will ensure that you have a copy of the data before deleting. If you do not execute them on two separate conditional steps, deletion occurs simultaneously with copy without any guarantee that the copy process was successful.

## **6.3.4 Recovery**

The OPERLOG is operationally independent of SYSLOG; that is, an installation can choose to run with either or both. A failure of the OPERLOG when the OPERLOG is active and SYSLOG (or the hardcopy message log) is not active causes SYSLOG to be activated automatically in an effort to reduce the impact to the operations and problem determination tasks. Other systems that were not affected by the failure will continue to write to the OPERLOG.

## **6.3.5 Performance**

After your initial log stream allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments are necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70-78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70–78 records

Operlog is a funnel type log stream: this means that we will expect offloads to be created as result of the OPERLOG process. There are some fields in the SMF88 that you might want to check against this log stream: DASD SHIFT, STRUCTURE FULL, ENTRY FULL and SC1/2/3. Refer to 8.6.3, “SMF Type 88 records and IXGRPT1 program” on page 291 for a description of these fields and what to do in case of one of this condition happens in your installation.

## **6.4 Resource Recovery Services**

RRS uses five log streams that can be shared by multiple z/OS instances in a sysplex. It keeps track of the resource managers registered with RRS, the Units of Recovery and their state.

### **6.4.1 Functional description**

RRS log streams can be defined either coupling facility based or as DASDONLY log streams, mainly dependent on the workload configurations and on where the RRS resource managers are running. This is only true if those RMs want to participate in the same set of transactions. If RRS resource managers running on different systems in a sysplex participate in the same

transactions, then coupling facility log streams are required in order to share information across multiple RRS instances. Each system that wants to connect to the set of log streams needs access to the coupling facility and to the DASD on which the offload data set will reside.

On the other hand, if the RMs never participate in the same transactions, they could be placed in DASDONLY log streams on each system. This configuration can occur either in a single system sysplex with one RRS image, or in a sysplex where each image runs independent workloads from each other (not very common).

RRS images on different systems in the sysplex run independently. However, RRS images that are in the same logging group share the same log stream to keep track of the work. If a system in the same logging group fails, RRS on a different system in the same logging group within the sysplex can use the shared log streams to take over the failed system's work. This means that the CF log streams allow the failed Resource Managers to restart on a different RRS image and that RRS then uses the shared logs to takeover the work. The takeover happens as a result of the failed Resource Managers restart. The Resource Managers must restart before the transactions are taken over and assigned to another RRS image. The shared logs provide the installation with the choice of where the Resource Managers can restart.

If there is only one system connected to the structure where the log streams reside or the log streams are DASD-only, then no other system can take over the failed system's work. In this situation, any outstanding sync points are resolved when RRS restarts using the logging group and the resource managers are active within the logging group.

You can have multiple RRS logging groups in a sysplex with different workloads. Each RRS group has a different set of log streams. At start time, RRS knows which group it belongs to from its procedure, and it then connects to the log streams for the specific group. The groupname identifier is part of the log stream name so they can differentiate from each other. The default log groupname is the sysplex name.

Example where your installation might want an RRS log group being a subset of the systems in a sysplex:

- ▶ You can restart RRS with a different log group name to cause a cold start and keep the data in the old logs.
- ▶ More frequently, you can use different log group names to subdivide the sync point work in a sysplex. For example, use a different log group for a test system.

Table 6-1 lists the RRS log streams and the related contents.

*Table 6-1 RRS Log streams and their content*

Log stream type	Log stream name	Content
RRS archive log	ATR.lgname.ARCHIVE	Information about completed URs. This log is optional. See Note for further details.
RRS main UR state log	ATR.lgname.MAIN.UR	The state of active URs. RRS periodically moves this information into the RRS delayed UR state log when UR completion is delayed.
RRS resource manager data log	ATR.lgname.RM.DATA	Information about the resource managers using RRS services.
RRS delayed UR state log	ATR.lgname.DELAYED.UR	The state of active URs, when UR completion is delayed.

Log stream type	Log stream name	Content
RRS restart log	ATR.lgname.RESTART	Information about incomplete URs needed during restart. This information enables a functioning RRS instance to take over incomplete work left over from an RRS instance that failed.

**Note:** The ARCHIVE log stream is an optional log stream and your installation can decide to use this log stream based on the following needs:

- ▶ If you need information to manually resolve units of recovery in catastrophic situations where RRS loses information about 'non completed' unit of recovery, for example situation where RRS loses its own log data and it cannot recover UR. You can access the ARCHIVE log stream to see which URs have completed so you can avoid processing completed transactions again.
- ▶ If your installation needs to maintain a history of completed transactions, that is, for accounting.

If you choose not to use the ARCHIVE log stream, a warning message is issued at RRS start up time about not being able to connect to the log stream, but RRS continues its initialization process.

### Criticality/persistence of data

RRS keeps information about units of recovery and their process in the log stream for the registered resource managers. This allows the coordination of the two phase commit protocol across multiple resource managers. For this reason, RRS data are very critical to the installation and it is not easy to deal with loss of data related to RRS log streams.

The critical piece of data for RRS is contained in the RM.DATA log stream. It is strongly recommended that you consider unconditionally duplexing the resource manager data log (ATR.lgname.RM.DATA). This log is small and infrequently updated, so its impact on performance is minimal, but any loss of data, unresolved gap, or any permanent error will force an RRS cold start

Since duplexing the logs can impact performance, it is suggested to conditional duplex all the other log streams as RRS is capable to tolerate unresolved gap in case the log stream suffers damage, although RRS will not be able to recover the lost data. (For a description of the meaning and consequence of cold start in an RRS environment, refer to *Systems Programmers guide to: RRS, SG24-6980.*)

### Log stream sizing

Table 6-2 on page 231 provides initial consideration on the amount of storage required for the RRS log streams. These recommendations should result in reasonably efficient usage of coupling facility storage while minimizing the likelihood that you will have to redefine the structures due to the variations in your workload. However, the exact amount of storage you need for each log stream depends on the installation RRS workload. SMF88 data can be used to understand if the structure sizes require any adjustments. If your RRS log streams are DASD ONLY log streams, determine the size of the Coupling Facility structure that would be needed for each log stream and make the staging data set as big as the structure would be.



After your initial set up, you can use the values from the SMF88 report as input to the Web-based tool to determine the exact amount of coupling facility space that you should allocate for a coupling facility structure. This tool can be accessed via the IBM Web site:

<http://www.ibm.com/servers/eserver/zseries/ps>

Because of the differences in activity rate among the multiple RRS log streams and also the possibility to easily manage their coupling facility storage sizing, it is suggested to map each coupling facility log stream to a unique coupling facility structure.

In case your installation has a constraint in the available number of coupling facility structures in your CFRM policy, you can think about grouping multiple RRS log stream in a single coupling facility structure. In this configuration:

- ▶ Re-evaluate the need for the ARCHIVE log stream in your installation and if your installation doesn't not make any use of the log stream data, delete the log stream. Before deleting the log stream, verify this action with all the resource managers to make sure they do not take advantage of this data in normal operations.
- ▶ Combine the RM.DATA and RESTART log streams in the same structure.
- ▶ Combine the MAIN.UR and DELAYED.UR log stream in the same structure.

Table 6-2 RRS log streams and allocation considerations

Log stream	Log stream name	Interim storage requirement	Maximum size in K	Initial size in K
Archive log	ARCHIVE	High	24576	6144
Resource Manager data log	RM.DATA	Low, if few resource managers; Medium, if many resource managers	4096	1024
Main UR state log	MAIN.UR	Medium	4096	1024
Restart log	RESTART	Medium	28672	9216
Delayed UR state log	DELAYED.UR	High	24576	8192

RRS log streams belong to the active log stream with the exception of the ARCHIVE. RRS has a trimming mechanism, so it keeps the content of the log streams up the date with the current workload running on the system. As for the ARCHIVE, this is a funnel-type log stream, where RRS writes a record to the log stream for each completed transaction. RRS just writes and never reads the ARCHIVE log and therefore never deletes archive log records. You should use a combination of AUTODELETE and RETPD to manage this log stream.

## 6.4.2 Definition

### Subsystem definition

There are no actual definitions in the RRS subsystem to map the log stream. Each member of a RRS group connects to the group's own set of log streams at start up time and the log stream are structured with a fixed name, that is, *ATR.groupname.DELAYED.UR* where the plexname is the sysplex name by default or the logging group name if multiple RRS logging group are present within the same sysplex. Either the plexname or the logging group is the connection between the RRS group and the set of log streams to be used.

If there is only one RRS logging group, then it is common to use the sysplex name as qualifier and nothing has to be done on the subsystem side to identify the log streams to connect to. If multiple RRS log groups exist within the sysplex, then the RRS procedure *GNAME* parameter need to be updated with the value of the specific qualifier of this RRS log group so this particular instance of RRS connects to this set of log streams peculiar to this logging group.

## Logger definition

RRS can use either Coupling Facility log streams or DASDONLY log streams. In an RRS environment, it is rare to see a mixing of Coupling Facility log streams and DASDONLY log streams. Usually the choice toward one media or another is mainly dictated by the capability of sharing the log streams across multiple instance of RRS.

## Coupling Facility log streams

The following samples show definitions for the RRS log stream and their logger view of the structures. To define a log stream, a combination of a structure and log stream definition are needed. The parameters and values that can affect performance and functionality are marked and discussed in the following sections.

### ***Structure definitions in the CFRM Policy for the RRS log streams***

This task is only required if you are planning to use Coupling Facility log streams. Before running any LOGR policy, you have to make sure that the correspondent Coupling Facility structure has already been defined in the CFRM policy and that the CFRM policy has been activated in the sysplex through the SETXCF command.

Here is a set of initial samples to define the Coupling Facility structures for the RRS log streams. Usually these sizing values are good for most of the installation, at least as starting point. We suggest that you use both RMF Coupling Facility report and the SMF88 output to validate your log stream configuration.

*Example 6-17 Structure Definition for MAIN.UR sample*

---

```
//MAINSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(RRS_MAINUR_1)
  SIZE(4096)
  INITSIZE(1024)
  PREFLIST(FACIL01, FACIL02)
```

---

*Example 6-18 Structure Definition for DELAYED.UR sample*

---

```
//DELSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(RRS_DELAYED_1)
  SIZE(24576)
  INITSIZE(8192)
  PREFLIST(FACIL01, FACIL02)
```

---

*Example 6-19 Structure Definition for ARCHIVE sample*

---

```
//ARCSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM) REPORT(YES)
  STRUCTURE NAME(RRS_ARCHIVE_1)
  SIZE(24576)
  INITSIZE(6144)
  PREFLIST(FACIL01, FACIL02)
```

---

*Example 6-20 Structure Definition for RM.DATA sample*

---

```
//RMSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM) REPORT(YES)
  STRUCTURE NAME(RRS_RMDATA_1)
  SIZE(4096)
  INITSIZE(1024)
  PREFLIST(FACIL01, FACIL02)
```

---

*Example 6-21 Structure Definition for RESTART sample*

---

```
//RSTSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(RRS_RESTART_1)
  SIZE(28672)
  INITSIZE(9216)
  PREFLIST(FACIL01, FACIL02)
```

---

**Logger structure definitions**

The structure used for the log stream affects performance. Key factors include the number and characteristics of the log streams which shares the structure. Best practise has shown that the best performance is achieved when in the RRS environment there is one log stream per structure.

Below are the examples for the definition of the structure in the LOGR policy.

*Example 6-22 MAIN.UR structure definition sample*

---

```
//MAINSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR)
  DEFINE STRUCTURE NAME(RRS_MAINUR_1)
  LOGSNUM(1)
  AVGBUFSIZE(158)
  MAXBUFSIZE(65276)
```

---

*Example 6-23 DELAYED.UR structure definition sample*

---

```
//DELAYSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  DEFINE STRUCTURE NAME(RRS_DELAYED_1)
    LOGSNUM(1)
    AVGBUFSIZE(158)
    MAXBUFSIZE(65276)
```

---

*Example 6-24 ARCHIVE structure definition sample*

---

```
//ARCHSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  DEFINE STRUCTURE NAME(RRS_ARCHIVE_1)
    LOGSNUM(1)
    AVGBUFSIZE(262)
    MAXBUFSIZE(65276)
```

---

*Example 6-25 RM.DATA structure definition sample*

---

```
//RMSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  DEFINE STRUCTURE NAME(RRS_RMDATA_1)
    LOGSNUM(1)
    AVGBUFSIZE(252)
    MAXBUFSIZE(1024)
```

---

*Example 6-26 RESTART structure definition sample*

---

```
//RSTSTR JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  DEFINE STRUCTURE NAME(RRS_RESTART_1)
    LOGSNUM(1)
    AVGBUFSIZE(158)
    MAXBUFSIZE(65276)
```

---

MAXBUFSIZE: MAXBUFSIZE in conjunction with AVGBUFSIZE is used to determine the CF structure ENTRY/ELEMENT ratio. When data is written to the CF, it is written in increments equal to ELEMENT size. A MAXBUFSIZE greater than 65276 gives an element size of 512; a MAXBUFSIZE equal to or less than 65276 results in an element size of 256.

With OS/390 R1.3, System Logger dynamically adjusts the Entry/Element ratio avoiding potential problem, specially if you are not planning to share the same structure between multiple log stream with in theory different characteristics.

## **Log stream definitions**

*Example 6-27 MAIN.UR log stream*

---

```
//MAINLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.MAIN.UR)
  STRUCTNAME(RRS_MAINUR_1)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_SIZE()
  LOWOFFLOAD(60)
  HIGHOFFLOAD(80)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  RETPD(0)
  AUTODELETE(NO)
```

---

*Example 6-28 DELAYED.UR log stream*

---

```
//DELLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.DELAYED.UR)
  STRUCTNAME(RRS_DELAYEDUR_1)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_DATACLAS()
  STG_SIZE(9000)
  LOWOFFLOAD(60)
  HIGHOFFLOAD(80)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  RETPD(0)
  AUTODELETE(NO)
```

---

*Example 6-29 ARCHIVE log stream*

---

```
//ARCHLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.ARCHIVE.UR)
  STRUCTNAME(RRS_ARCHIVE_1)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_SIZE(9000)
  LOWOFFLOAD(0)
  HIGHOFFLOAD(80)
  STG_DUPLEX(NO)
  RETPD(2)
```

AUTODELETE(YES)

---

*Example 6-30 RM.DATA log stream*

---

```
//RMLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.RM.DATA)
  STRUCTNAME(RRS_DATA_1)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_DATACLAS()
  STG_SIZE(9000)
  LOWOFFLOAD(60)
  HIGHOFFLOAD(80)
  STG_DUPLEX(YES)
  DUPLEXMODE(UNCOND)
  RETPD(0)
  AUTODELETE(NO)
```

---

*Example 6-31 RESTART log stream*

---

```
//RESTLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.RESTART)
  STRUCTNAME(RRS_RESTART_1)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_DATACLAS()
  STG_SIZE(9000)
  LOWOFFLOAD(60)
  HIGHOFFLOAD(80)
  STG_DUPLEX(YES)
  DUPLEXMODE(COND)
  RETPD(0)
  AUTODELETE(NO)
```

---

**AUTODELETE and RETPD:** AUTODELETE and RETPD can have a disastrous effect on all the RRS log stream except the ARCHIVE if specified other than AUTODELETE(NO) and RETPD(0). With AUTODELETE(YES) and RETPD>0, even though RRS will attempt to delete un-necessary log entries, all data will be off-loaded to the offload data sets and held for the number of days specified for RETPD. AUTODELETE(YES) allows the System Logger (rather than RRS) to decide when to delete the data. When a new offload data set is allocated and AUTODELETE(YES) is specified, the System Logger will delete the data on the old offload data set. Since data on the MAIn.UR are managed by RRS, you must let RRS managed the life of the records on this log streams so there will not be the risk that RRS will need records that have been deleted by logger because of the AUTODELETE option.

For the ARCHIVE log stream, RRS never uses information written on the Archive log; the information is intended for the installation to use if a catastrophic problem occurs. You must use retention period and autodelete support to delete old entries - specify these value big

enough to manage this log stream on a reasonable size and to provide enough coverage in time to recover any potential situation.

**HIGHOFFLOAD:** The HIGHOFFLOAD parameter is used to determine when the space dedicated to the log stream in the Coupling Facility is filling up and an offload need to be initiated to re-gain available space. HIGHOFFLOAD should be set at 80% for all the RRS log streams at least initially and then use the SMF88 report to evaluate if this value need same tuning.

**LOWOFFLOAD:** The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. In the RRS environment, the LOWOFFLOAD value should be high enough to retain the data required for backout of the UR but low enough to keep the number of offloads to a minimum.

LOWOFFLOAD should be set between 40 and 60% for all the RRS log stream as described in the examples and 0% for the ARCHIVE.

**LS\_SIZE:** LS\_SIZE defines the allocation size for the *offload* data sets. It should be specified large enough to contain several offloads, possibly a day's worth. All RRS log streams except ARCHIVE should only offload a minimal amount of data.

**Caution:** If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member. The default value in ALLOCxx is two tracks. Refer to the *z/OS MVS Initialization and Tuning Reference, SA22-7592*.

It is very important to remember log stream staging and offload (log) data sets are single extent VSAM linear data sets, and the shareoptions *must* be specified as '3,3'. If the shareoptions are anything other than '3,3' there is a risk the System Logger will be unable to read offloaded data and provide RRS with return code 8, reason code 804 (IlgRsnCodeNoBlock).

**STG\_SIZE:** For a Coupling Facility log stream, STG\_SIZE defines the size of the staging data set to be allocated if STG\_DUPLEX(YES). If DUPLEXMODE(UNCOND) is specified the data in the Coupling Facility log stream will always be duplexed to the staging data set. If DUPLEXMODE(COND) is specified the data in the Coupling Facility log stream is duplexed to the staging data set only if the CF becomes volatile or failure dependent.

The size of the staging data set (STG\_SIZE) must be large enough to hold as much data as the log stream storage in the Coupling Facility. IF STG\_SIZE is (), logger allocates a staging data set large enough to hold the coupling facility structure size.

Data is written to the staging data set in 4096 byte increments, regardless of the buffer size.

**STG\_DUPLEX:** STG\_DUPLEX(YES) with DUPLEXMODE(COND) means that if the CF becomes volatile, or resides in the same failure domain as the System Logger system, the log stream data will be duplexed to the staging data set; otherwise it is duplexed to buffers in the System Logger data space. A CF is in the same failure domain when the Coupling Facility LPAR and the LPAR running this OS/390 or z/OS reside in the same physical hardware box, central processing complex (CPC). Duplexing to the staging data set means the cost of an I/O will be incurred for each write.

**LS\_DATACLAS:** With this keyword, you can specify the SMS Dataclass that is used for the allocation of offload data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure the desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the share options.

Offload data sets may use a CI size up to 24K; in general, the larger CI size provides better performance, however, if the log stream consists mostly of log blocks of a much smaller size, the 24-K CI size may be wasteful.

**Attention:** For a shared Coupling Facility log stream which has connections to multiple z/OS images when the offload process is triggered, the offload may be performed by the logger address space on a different z/OS image. If the offload requires the movement of data to the log (offload) data sets, and the current data set fills, a new data set will be allocated. Unless the shareoptions are specified 3,3 the logger address space on the z/OS image where RRS is running may not be able to access the data set.

**STG\_DATACLAS:** With this keyword, you can specify the SMS Dataclass that is used for the allocation of staging data sets. The Automatic Class Selection (ACS) routines can override this value so you should ensure that desired dataclass is used. In addition to the size of the data set, the dataclass defines the CI Size for the data set along with the share options. Staging data sets must use a CI size of 4 K.

**Important:** the data class is the only way to specify the share options for a data set. You **MUST** ensure that all System Logger data sets, including staging data sets, are allocated with a dataclass that defines share options (3,3).

### DASD-only log stream definitions

The following samples show definitions for the RRS log stream when defined as DASD-only log streams. The parameters and values that can affect performance and functionality are marked and discussed in the following sections.

*Example 6-32 MAIN.UR DASD-only log stream sample*

---

```
//MAINLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
  DEFINE LOGSTREAM NAME(ATR.groupname.MAIN.UR)
    DASDONLY(YES)
    LS_DATACLAS(LOGR4K)
    HLQ(IXGLOGR)
    LS_SIZE(1024)
    STG_DATACLAS()
    STG_SIZE(9000)
    LOWOFFLOAD(60)
    HIGHOFFLOAD(80)
    RETPD(0)
    AUTODELETE(NO)
    DASDONLY(YES)
    MAXBUFSIZE(65532)
```

---

*Example 6-33 DELAYED.UR DASD-only log stream sample*

---

```
//DELLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
  DEFINE LOGSTREAM NAME(ATR.groupname.DELAYED.UR)
    DASDONLY(YES)
    LS_DATACLAS(LOGR4K)
```

---



```
HLQ(IXGLOGR)
LS_SIZE(1024)
STG_DATACLAS()
STG_SIZE(9000)
LOWOFFLOAD(60)
HIGHOFFLOAD(80)
RETPD(0)
AUTODELETE(NO)
DASDONLY(YES)
MAXBUFSIZE(65532)
```

---

*Example 6-34 ARCHIVE DASD-only log stream sample*

---

```
//ARCHLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.ARCHIVE.UR)
DASDONLY(YES)
LS_DATACLAS(LOGR4K)
HLQ(IXGLOGR)
LS_SIZE(1024)
STG_DATACLAS()
STG_SIZE(9000)
LOWOFFLOAD(0)
HIGHOFFLOAD(80)
RETPD(15)
AUTODELETE(YES)
DASDONLY(YES)
MAXBUFSIZE(65532)
```

---

*Example 6-35 RM.DATA DASD-only log stream sample*

---

```
//RMLLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(ATR.groupname.RM.DATA)
DASDONLY(YES)
LS_DATACLAS(LOGR4K)
HLQ(IXGLOGR)
LS_SIZE(1024)
STG_DATACLAS()
STG_SIZE(9000)
LOWOFFLOAD(60)
HIGHOFFLOAD(80)
RETPD(0)
AUTODELETE(NO)
DASDONLY(YES)
MAXBUFSIZE(65532)
```

---

*Example 6-36 RESTART DASD-only log stream sample*

---

```
//RESTLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DATA TYPE(LOGR)
```

```

DEFINE LOGSTREAM NAME(ATR.groupname.RESTART)
  DASDONLY(YES)
  LS_DATACLAS(LOGR4K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  STG_DATACLAS()
  STG_SIZE(9000)
  LOWOFFLOAD(60)
  HIGHOFFLOAD(80)
  RETPD(0)
  AUTODELETE(NO)
  DASDONLY(YES)
  MAXBUFSIZE(65532)

```

---

**MAXBUFSIZE:** MAXBUFSIZE may be specified for a DASDONLY log stream, it defines the largest block that can be written to the log stream. The default value is 65532 and should be used in a RRS environment.

**STG\_DUPLEX:** STG\_DUPLEX(YES) with DUPLEXMODE(UNCOND) are implicitly defaulted in DASDONLY log streams.

### Security definitions

The RRS log streams can be protected in the *Security Access Facility (SAF)* LOGSTRM class. To enable the protection, you need to give READ access to the TSO user IDs or groups that will need to retrieve log data through the RRS panels and the UPDATE access level to the RRS address space user ID.

If your installation does not require this level of security, you can define the log stream with READ as universal access. This will allow all the user ID in your installation to browse the RRS log streams data.

#### *Example 6-37 Sample log stream security definitions*

---

```

RDEFINE LOGSTRM ATR.plexname.RESTART UACC(NONE)
RDEFINE LOGSTRM ATR.plexname.MAIN.UR UACC(NONE)
RDEFINE LOGSTRM ATR.plexname.ARCHIVE UACC(NONE)
RDEFINE LOGSTRM ATR.plexname.DELAYED.UR UACC(NONE)
RDEFINE LOGSTRM ATR.plexname.RM.DATA UACC(NONE)
PERMIT ATR.plexname.RESTART CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
PERMIT ATR.plexname.MAIN.UR CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
PERMIT ATR.plexname.ARCHIVE CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
PERMIT ATR.plexname.DELAYED.UR CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
PERMIT ATR.plexname.RM.DATA CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
PERMIT ATR.plexname.RESTART CLASS(LOGSTRM) ID(tso_userid) ACCESS(READ)
PERMIT ATR.plexname.MAIN.UR CLASS(LOGSTRM) ID(tso_userid) ACCESS(READ)
PERMIT ATR.plexname.ARCHIVE CLASS(LOGSTRM) ID(tso_userid) ACCESS(READ)
PERMIT ATR.plexname.DELAYED.UR CLASS(LOGSTRM) ID(tso_userid) ACCESS(READ)
PERMIT ATR.plexname.RM.DATA CLASS(LOGSTRM) ID(tso_userid) ACCESS(READ)

SETROPTS CLASSACT(LOGSTRM)

```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

## 6.4.3 Operational considerations

In this section we discuss operation considerations.

### ***Application start up***

Every time an application that uses RRS starts, RRS needs to go through the log stream reading and updating the RM.DATA and reading the other log streams. This process needs to be performed sequentially, so forth, RRS enqueues on SYSZATR.plexname/logrp.RESTART resource. Bear in mind that the application start process can take some time and if you start multiple applications using RRS at the same time, there might be some delay.

### ***RRS shutdown***

Before shutting RRS down, you should shut down all the RRS applications. Once the RRS application are down, issue the following commands:

- ▶ SETRRS CANCEL command to shut down RRS.
- ▶ D LOGGER,L,LSN=ATR.\* and verify that the connections number is 0.

If the connection number is not 0, issue the D LOGGER,C,LSN=ATR.\* to verify the outstanding connection in every system in the sysplex. If the number of connections is zero, then RRS needs to go through log streams recovery.

### ***Power-on-Reset***

Make sure that RRS disconnects from its log streams before shutting down the z/OS images and performing a power-on-reset of the coupling facility and z/OS CECs. This action will prevent loss of data or message ATR212I RRS DETECTED LOG DATA LOSS at IPL time.

When you are shutting down your system, after all the resource managers are down, you can issue the SETRRS CANCEL command to close RRS and force it to disconnect from its log stream. This will allow logger to offload the data and harden them on the offload data set.

### ***Starting RRS with no log stream defined***

RRS is not able to complete its initialization if it cannot connect to the log stream. You will receive the following messages on the console and RRS will shut down.

*Example 6-38 RRS start up messages*

---

```
S RRS
ATR221I RRS IS JOINING RRS GROUP A ON SYSTEM #@$1
IXG231I IXGCONN REQUEST=CONNECT TO LOG STREAM ATR.A.RM.DATA DID NOT
SUCCEED FOR JOB RRS. RETURN CODE: 00000008 REASON CODE: 0000080B
DIAG1: 00000008 DIAG2: 0000F801 DIAG3: 05030004 DIAG4: 05020010
ATR130I RRS LOGSTREAM CONNECT HAS FAILED FOR
MANDATORY LOGSTREAM ATR.A.RM.DATA.
RC=00000008, RSN=00000000
IEA989I SLIP TRAP ID=X13E MATCHED. JOBNAME=RRS , ASID=0082.
ASA2013I RRS INITIALIZATION FAILED. COMPONENT ID=SCRRS
```

---

## **6.4.4 Recovery**

### ***Log streams recovery***

To perform RRS log stream recovery, restart and then shutdown all the RRS instances to force RRS to connect and then disconnect to the log streams. This will try to clear and solve any potential problem.

### **Cold start**

The most critical log stream is the RM.DATA. If, for any reason, data is lost, RRS cannot recover and the only way to resume is to perform a cold start. In case of a cold start, all the in-flight UOW for the Resource Managers that were registered with RRS will abend.

Although to perform a cold start you only need to redefine the RM.DATA log stream, it is suggested to delete all the current log streams, define new ones and restart RRS. This will clean all the previous information and perform a clean restart.

In case you want to keep the old logs for problem determination, then you only need to redefine the RM.DATA and use the other old log streams.

### **Performance**

After your initial log streams allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments are necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70-78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70–78 records

All the RRS log streams with the exception of the ARCHIVE are active log stream: this means that RRS takes care of maintaining currency of data within the log stream. For this reason, after an accurate tuning, while we should expect offloads for the ARCHIVE log stream, we should see that offload are eliminated or minimized for all the other log streams.

In addition, generally speaking, RRS log streams usually are not very critical from a performance perspective. There are some fields in the SMF88 that you might want to check against the active log streams to determine if they are tuned properly: DASD SHIFT, STRUCTURE FULL, ENTRY FULL and OFFLOAD.

As regards to the ARCHIVE log stream, being a funnel type log stream, offload are expected - for this reason you should check the fields DASD SHIFT, STRUCTURE FULL, ENTRY FULL and SC1/2/3 in the SMF88 for this particular log stream.

Refer to 8.6.3, “SMF Type 88 records and IXGRPT1 program” on page 291 for a description of these fields and what to do in case of one of this condition happens in your installation

## **6.5 System Automation for OS/390**

The following describe the log streams used by System Automation for OS/390 and their characteristics.

### **6.5.1 Functional description**

The Automation Manager optionally uses two log streams: the History log stream where it writes the work items that have been processed and the Message log stream where application message are written from the Automation Manager itself and the NetView® Agents.

In addition to these two log streams, APAR OW56107 introduced a third log stream, the health checker. The SYSOPS function, on interval base, performs checks in the systems and writes the results on this log stream for later retrieval.

These log streams are funnel type log stream where System Automation for OS/390 doesn't perform any trimming on the data but relies on the logger function to remove old records from the log stream. For this reason, you should use RETPD and AUTODELETE keyword on the log stream definition to avoid these log stream from growing unlimited.

### Criticality/persistence of data

System Automation for OS/390 uses these log streams to accumulate messages and information about the behavior of the systems to potentially trigger some event or reactions. They are funnel-type log streams, where System Automation writes and the agents can retrieve the data to understand what's happening. System Automation does not manage the records within the log streams: for this reason, you should use a combination of AUTODELETE and RETPD to manage these log streams and let logger delete the records after the period of time you decide you might need to keep the data for your installation.

In a multi systems sysplex, it is preferred to have these log streams defined as coupling facility based log streams because it is possible to record data and to view data for each system anywhere in the sysplex. In a monoplex or in a single system sysplex, DASD-only log stream can be a good alternative configuration.

### Log stream sizing

The initial recommendation on how to size the System Automation for OS/390 interim storage is that you can start with two structures, each with a size of about 8MB in a Coupling Facility or a 2000 block if you are using DASD-only log stream and use SMF88 to verify how accurate this configuration is for your installation.

One structure should be shared between the History log stream and the Message log and the other dedicated to the Health Checker log stream. If you are not constraint, you can also dedicate a coupling facility structure to each log stream.

## 6.5.2 Definition

### Subsystem definitions

There are no definition at subsystem levels since the name of the log streams are fixed. When System Automation for OS/390 initializes, it connects to these log streams.

The names of the log streams are:

HSA.MESSAGE.LOG, HSA.WORKITEM.HISTORY and ING.HEALTH.CHECKER.HISTORY

### Structure definition in the CFRM Policy for the SAFO for OS/390

*Example 6-39 Structure definitions sample*

---

```
//SAFOLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(HSA_LOG)
    SIZE(16384)
    INITSIZE(8192)
    FULLTHRESHOLD(0)
    PREFLIST(FACIL02, FACIL01)

  STRUCTURE NAME(ING_HIST)
    SIZE(16384)
    INITSIZE(8192)
```

```
FULLTHRESHOLD(0)
PREFLIST(FACIL02, FACIL01)
```

---

## Logger definitions

Example 6-40 shows how to define the log streams in the LOGR policy.

*Example 6-40 System Automation for OS/390 log stream sample*

---

```
//SAFOLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR)

DEFINE STRUCTURE NAME(HSA_LOG)
  LOGSNUM(2)
  MAXBUFSIZE(65532)
  AVGBUFSIZE(1024)

DEFINE STRUCTURE NAME(ING_HIST)
  LOGSNUM(1)
  MAXBUFSIZE(65532)
  AVGBUFSIZE(1024)

DEFINE LOGSTREAM NAME(HSA.MESSAGE.LOG)
  STRUCTNAME(HSA_LOG)
  LS_DATACLAS(LOGR24K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  LOWOFFLOAD(0)
  HIGHOFFLOAD(80)
  STG_DUPLEX(NO)
  RETPD(7)
  AUTODELETE(YES)
  OFFLOADRECALL(YES)
  DASDONLY(NO)

DEFINE LOGSTREAM NAME(HSA.WORKITEM.HISTORY)
  STRUCTNAME(HSA_LOG)
  LS_DATACLAS(LOGR24K)
  HLQ(IXGLOGR)
  LS_SIZE(1024)
  LOWOFFLOAD(0)
  HIGHOFFLOAD(80)
  STG_DUPLEX(NO)
  RETPD(7)
  AUTODELETE(YES)
  OFFLOADRECALL(YES)

DEFINE LOGSTREAM NAME(ING.HEALTH.CHECKER.HISTORY)
  STRUCTNAME(ING_HIST)
  STG_DUPLEX(NO)
  LS_DATACLAS(LOGR24K)
  LS_SIZE(4096)
  AUTODELETE(YES)
  RETPD(365)
  HLQ(IXGLOGR)
  HIGHOFFLOAD(80)
```

LOWOFFLOAD(0)  
DIAG(YES)

---

## Security definitions

You can protect your log streams defining them as resources in the *Security Access Facility* (SAF) LOGSTRM class. The NetView agents as well as the automation manager address spaces need UPDATE access to the log streams.

*Example 6-41 Sample log streams security definitions*

---

```
RDEFINE LOGSTRM ING.HEALTH.CHECKER.HISTORY UACC(READ)
RDEFINE LOGSTRM HSA.WORKITEM.HISTORY UACC(READ)
RDEFINE LOGSTRM HSA.MESSAGE.LOG UACC(READ)
PERMIT ING.HEALTH.CHECK.LOG CLASS(LOGSTRM) ID(Automatin_Manager_userid) ACCESS(UPDATE)
PERMIT HSA.WORKITEM.HISTORY CLASS(LOGSTRM) ID(Automation_Manager_userid) ACCESS(UPDATE)
PERMIT HSA.MESSAGE.LOG CLASS(LOGSTRM) ID(Automation_Manager_userid) ACCESS(UPDATE)
PERMIT ING.HEALTH.CHECK.LOG CLASS(LOGSTRM) ID(Netview_Agent_userid) ACCESS(UPDATE)
PERMIT HSA.WORKITEM.HISTORY CLASS(LOGSTRM) ID(Netview_Agent_userid) ACCESS(UPDATE)
PERMIT HSA.MESSAGE.LOG CLASS(LOGSTRM) ID(Netview_Agent_userid) ACCESS(UPDATE)
```

---

## 6.5.3 Operational considerations

### *Start up problem*

If System Automation for OS/390 log streams are not defined or are defined not correctly in the logger policy, you can expect the following message at start up time. SA/390 will initialize but the functions that will need the log streams to operate will not be functional. See Example 6-42.

*Example 6-42 MSOPS start up messages*

---

```
IXG231I IXGCONN REQUEST=CONNECT TO LOG STREAM
ING.HEALTH.CHECKER.HISTORY DID NOT SUCCEED FOR JOB MSOPS1. RETURN
CODE: 00000008 REASON CODE: 0000080B DIAG1: 00000008 DIAG2:
0000F801 DIAG3: 05030004 DIAG4: 05020010
HSAM5403I STREAM ING.HEALTH.CHECKER.HISTORY IS NOT DEFINED IN THE LOGR POLICY.
IXG231I IXGCONN REQUEST=CONNECT TO LOG STREAM HSA.MESSAGE.LOG DID NOT
SUCCEED FOR JOB MSOPS1. RETURN CODE: 00000008 REASON CODE: 0000080B
DIAG1: 00000008 DIAG2: 0000F801 DIAG3: 05030004 DIAG4: 05020010
HSAM5403I STREAM HSA.MESSAGE.LOG IS NOT DEFINED IN THE LOGR POLICY.
IXG231I IXGCONN REQUEST=CONNECT TO LOG STREAM HSA.WORKITEM.HISTORY DID
NOT SUCCEED FOR JOB MSOPS1. RETURN CODE: 00000008 REASON CODE:
0000080B DIAG1: 00000008 DIAG2: 0000F801 DIAG3: 05030004 DIAG4:
05020010
HSAM5403I STREAM HSA.WORKITEM.HISTORY IS NOT DEFINED IN THE LOGR POLICY
```

---

## 6.5.4 Recovery

### *Data gap in log streams*

When System Automation for OS/390 detects a gap in the sequence of logger records, it logically deletes all the data following this gap regardless of the retention period. For example, this can happen when an user manually deletes some logger data sets or offload data sets are allocated with incorrect shareoptions.

## 6.5.5 Performance

Usually these log streams do not present many concerns from a performance perspective toward the System Automation for OS/390 log stream. After your initial log stream allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments is necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70–78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70–78 records

Refer to 8.6.4, “IXGRPT1 Field Summary and IXGSMF88 Cross Reference” on page 293 on how to use the IXGRPT1 output.

## 6.6 WebSphere Application Server

WebSphere for z/OS is structured as a multiple address spaces application: it has a control region where the authorized code can run and multiple server regions where the application code runs. These server regions are dynamically started based on the amount of work by WLM services. All of these regions use a log stream to accumulate error messages. Each region can have its own log stream or can share the log stream with other servers.

### 6.6.1 Functional description

The WebSphere for z/OS error log is a log stream that contains error messages from the Application Server. These messages are typically more detailed than you would see on the system console or in the job output of the server's address space. For example, you might see a message in the joblog of an application server saying that naming registration has failed, while the error log is more likely to have a message indicating why it failed.

#### Criticality/persistence of data

WebSphere Application Server is using this error log stream to accumulate error messages. It is a funnel-type log stream, where WebSphere Application Server just writes and never reads the log back; therefore never deletes archive log records. For this reason, you should use a combination of AUTODELETE and RETPD to manage this log stream.

Depending on the WebSphere for z/OS configuration, you can either use Coupling Facility or Dasd-only for this log stream. Dasd-only log stream can only be either dedicated to a single instance of WebSphere for z/OS or shared across multiple instances of WebSphere for z/OS running on the same z/OS image. Coupling facility based log stream is needed if you want a single merged log stream.

#### Log stream sizing

The WebSphere for z/OS log stream is a funnel-type log stream and the size mainly depends on the amount of error messages the WebSphere Application Server generates. Initially, you can start with a log stream of about 8MB in a Coupling Facility or a 2000 block if you are using DASD-only log stream and use SMF88 to verify how accurate this configuration is for your installation.



In case you have a spike in the messages rate you are writing on the log stream, your application should not suffer any trouble since your data will be offloaded into the offload data sets.

## 6.6.2 Definition

### Subsystem definition

WebSphere for z/OS manages its environment data through the Administration application and writes the environmental data into the system management database. To add or change environment variable data, you must enter environment data pairs (an environment variable name and its value) on the sysplex, server, or server instance properties form. When you activate a conversation, the environment variable data is written to HFS files. WebSphere for z/OS determines which values are the most specific for an environment file. For instance, a setting for a server instance takes precedence over the setting for the same variable for its server, and a setting for a server takes precedence over the setting for the same variable for its sysplex. If you modify an environment file directly and not through the Administration application, any changes are overwritten when you activate a conversation or prepare for a cold start.

When you activate a conversation, WebSphere for z/OS writes the environment data to an HFS file for each server instance. The path and name for each environment file is:

```
cb-pathname/controlinfo/envfile/SYSPLEX/SRVNAME/current.env
```

One of the variables in the `current.env` file defines the log stream that this WebSphere for z/OS instance will use. The name of the error log is specified using the environment variable:

```
LOGSTREAMNAME=<LOG_STREAM_NAME>
```

- ▶ The log stream can be defined at *Sysplex level* and so all servers in the sysplex will share the log stream and write their messages unless otherwise specified.
- ▶ Or it can be defined at *Server level* and so forth all the server instances under this server share and write to this log stream. This overrides the sysplex value for this particular server.
- ▶ Or it can be specified at *Server Instance* for the single instance running on a specific system to write to this particular log stream.

### Logger definition

The definition for the WebSphere for z/OS log stream can be obtain through the customization dialog. During the installation task, one of the job that are created by the installation clist is to define the log stream.

The user initiates the installation task by calling the clist via '`ex hlq.sbboclib(bbowstr) options`'

As a result, the user is presented with a menu to define resources. One of the tasks on the main menu is to input the variable. If you select the task 'Define Variable' (option 2), you are presented with 6 choices where you can enter all the variables to be used later in the customization jobs.

If you select the “WebSphere Customization” section (Option 2) the user is requested to input the variable that will be used to define the log stream. Figure 6-3 is a sample of the ISPF dialog that will be presented to the user.

```

----- WebSphere for z/OS Customization -----
Option  ===>
WebSphere Customization (2 of 4)
WebSphere Error Logstream Information
Name.....: BBO.BOSSERRS
Data class ..: LOGGER
Storage class.....: LOGPLEXU
HLQ for data sets....: LOGPLEXU

Is logstream CF resident (Y|N) : N
If yes, specify structure name.: -
If no, specify: logstream size.: 900
                  staging size...: 900

RRS Logstream Information
Group name.....: PELPLEXU
Data class.....: LOGGER
Storage class.....: LOGPLEXU
HLQ for data sets....: LOGPLEXU

Is logstream CF resident (Y|N)..: Y
Create RRS PROC (Y|N).....: N
M& b 15/0

```

Figure 6-3 Sample ISPF dialog to input variables to define log stream

Where:

- ▶ Name: Name of your WebSphere for z/OS error log stream that will be created.
- ▶ Data class: An existing DFSMS data class for the log stream data set allocation.
- ▶ Storage class: An existing DFSMS storage class for allocation of the DASD staging data set for this log stream.
- ▶ HLQ for data sets: The high-level qualifier for your log stream data set name and staging data set name that will be created.
- ▶ Is logstream CF resident (YIN): If you want the log stream to be created on a coupling facility, specify “Y”. If on DASD, specify “N”.
- ▶ If yes, specify structure name: If using the coupling facility, specify the coupling facility structure to be used for the log stream. Only valid if you specified Y on the previous entry.
- ▶ If no, specify: logstream size: Specifies the size, in 4K blocks, of the log stream DASD data sets for the log stream being defined.
- ▶ If no, specify: staging size: Specifies the size, in 4K blocks, of the DASD staging data set for the log stream being defined.

For detailed information about setting up the error log stream (including how to define the data sets, and give proper authority to server identities so they can write to it), see WebSphere Application Server for z/OS V4.0.1: Installation and Customization, GA22-7834.

After you have completed to input your variables customization job are generated in your *hlq.data\_set.CNTL*. The member BBOERRLG contains the definition for the log stream.

If you chose to use a Coupling Facility log stream, remember that you need to define the structure to the CFRM policy first.

## Definition in the CFRM Policy for the WebSphere for z/OS log stream

This task is only required if you are planning to use Coupling Facility log stream. Before running any LOGR policy, you have to make sure that the correspondent Coupling Facility structure has already been defined in the CFRM policy and that the CFRM policy has been activated in the sysplex through the SETXCF command.

Here is a set of initial samples to define the Coupling Facility structures for the WebSphere for z/OS log stream. You can use the following value as initial size and then use both RMF Coupling Facility report and the SMF88 output to validate your log stream configuration.

### *Example 6-43 Sample of CFRM definitions for the WebSphere log stream structure*

---

```
//WASLOG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(CFRM)
  STRUCTURE NAME(WAS)
  SIZE(8192)
  PREFLIST(FACIL01,FACIL02)
```

---

## Coupling Facility log streams definitions

The following sample shows the content of the BBOERRLG member for a Coupling Facility log stream. To define a log stream, a combination of a structure and log stream definition are needed. The parameters and values that can affect performance and functionality are marked and discussed in the following section.

### *Example 6-44 BBOERRLG sample*

---

```
//BBOERRLG JOB CLASS=A,MSGCLASS=A
//POLICY EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DATA TYPE(LOGR)
  DEFINE STRUCTURE NAME(WAS)
    LOGSNUM(1)
    MAXBUFSIZE(4096)

  DFINE LOGSTREAM NAME(BBO.BOSSERRS)
    STRUCTNAME(WAS)
    HLQ(LOGPLEX)
    AUTODELETE(YES)
    RETPD(1)
    LS_DATACLAS(LOGGER)
    LS_STORCLAS(LOGPLEX)
```

---

**MAXBUFSIZE:** MAXBUFSIZE in conjunction with AVGBUFSIZE is used to determine the CF structure ENTRY/ELEMENT ratio. In this case, since AVGBUFSIZE is not specified, it will be half the amount of MAXBUFSIZE value. When data is written to the CF, it is written in increments equal to ELEMENT size. A MAXBUFSIZE greater than 65276 gives an element size of 512; a MAXBUFSIZE equal to or less than 65276 results in an element size of 256.

Starting with OS/390 R1.3, System Logger dynamically adjusts the Entry/Element ratio avoiding potential problem, specially if you are not planning to share the same structure between multiple log stream with in theory different characteristics.

AUTODELETE and RETPD: WebSphere for z/OS never uses the information written on the error log: they are only for debugging scope. For this reason, to avoid a never ending accumulation of data, you must use retention period and autodelete support to delete old entries - specify these value big enough to manage this log stream on a reasonable size and to provide enough coverage in case you need to perform any debugging.

HIGHOFFLOAD: The HIGHOFFLOAD parameter is used to determine when the space dedicated to the log stream in the interim device is filling up and an offload need to be initiated to re-gain available space. HIGHOFFLOAD should be set at 80% for the error log stream at least initially and then use the SMF88 report to evaluate if this value need same tuning.

LOWOFFLOAD: The LOWOFFLOAD value defines the amount of data which may be retained in the log stream interim storage following an offload process. Since WebSphere for z/OS never reuses the data written in the lo stream, there is no point to keep any data in the log stream. For this reason, the LOWOFFLOAD value should be 0 to force all the data out of the interim device toward the offload data set.

LS\_SIZE: LS\_SIZE defines the allocation size for the offload data sets. It should be specified large enough to contain several offloads, possibly a day's worth.

**Caution:** There is no LS\_SIZE value generated in the job as result of the customization dialog. Be sure that there is an allocation size for this data set specified in the associated SMS Dataclas. If LS\_SIZE is not specified in the log stream definition, and an extent size is not specified in the data class pointed to by LS\_DATACLAS, the value is taken from the ALLOCxx Parmlib member or set via an ACS (Automatic Class Selection) routine. The default value in ALLOCxx is 2 tracks. Refer to the *z/OS MVS Initialization and Tuning Reference, SA22-7592*.

It is very important to remember log stream staging and offload (log) data sets are single extent VSAM linear data sets, and the shareoptions MUST be specified as '3,3'.

### ***DASD-only log streams definitions***

The following sample shows the definition for the WebSphere for z/OS error log stream when defined as DASD-only log stream. The parameters and values that can affect performance and functionality are marked and discussed in the following section.

*Example 6-45 BBOERRLG DASD-only log stream sample*

---

```
//BBOERRLG JOB ,,MSGLEVEL=1,MSGCLASS=R,CLASS=A,REGION=OM,
//BBORCLGS EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR)
DEFINE LOGSTREAM NAME(BBO.BOSSERRS)
  DASDONLY(YES)
  HLQ(LOGPLEX)
  LS_SIZE(2000)
  STG_SIZE(2000)
  MAXBUFSIZE(4096)
  AUTODELETE(YES)
  RETPD(1)
  LS_DATACLAS(LOGGER)
  LS_STORCLAS(LOGPLEX)
```

---

MAXBUFSIZE: MAXBUFSIZE may be specified for a DASDONLY log stream, it defines the largest block that can be written to the log stream.

STG\_DUPLEX: STG\_DUPLEX(YES) with DUPLEXMODE(UNCOND) are implicitly default for DASDONLY log streams.

### Security definitions

The WebSphere for z/OS log stream can be protected in the *Security Access Facility (SAF)* LOGSTRM class. Each WebSphere for z/OS Application Server user ID must be permitted UPDATE access to the log stream.

*Example 6-46 Sample log stream security definition*

---

```
RDEFINE LOGSTRM BBO.BOSSERRS UACC(READ)
PERMIT BBO.BOSSERRS CLASS(LOGSTRM) ID(CBASR2) ACCESS(UPDATE)
SETROPTS CLASSACT(LOGSTRM)
```

---

The log stream name defined to RACF is the name specified on the NAME parameter of the DEFINE LOGSTREAM statement in the LOGR policy.

### 6.6.3 Operational consideration

There is no external control to dynamically enable or disable the log and also to filter the content of tracing.

The data stored is not in a readable format so WebSphere for z/OS provides a REXX EXEC (BBORBLOG) that allows you to browse the data contained in the error log stream. By default, BBORBLOG formats the error records to fit a 3270 display.

You can view the error log stream output using the BBORBLOG browser. To invoke the browser, go to ISPF option 6 and enter:

```
ex 'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX format option'
```

Where:

- ▶ **log stream name** is the name of the log stream.
- ▶ **format option** is the LRECL length of the log stream records.
  - **80** This is the default. The log stream record will be formatted on a LRECL length of 80 characters. Additional lines will be wrapped.
  - **NOFORMAT** This turns off formatting. The error log message appears as one log message string in the browse file.

In this example, BBORBLOG resides in BBO.SBBOEXEC and BBO.BOSSXXXX is the LOG\_STREAM\_NAME that was configured in the SMUI. The browser creates a data set named 'USERID.LOG\_STREAM\_NAME' which will contain the formatted contents of the log stream. When the browser is executed, it does the following:

1. Allocates a data set called USERID.LOG\_STREAM\_NAME, which overwrites any duplicate data sets.
2. Populates the data set with the contents of the log stream.
3. Puts the user in browse mode on the data set.

### 6.6.4 Recovery

There are no particular considerations from a recovery point of view for this log stream since there are no critical data in it.

During server initialization, WebSphere for z/OS makes an attempt to connect to the appropriate log stream. If this connection is successful, you will see the following message, which indicates the name of the log stream being used:

```
BOU0025I ERRORS WILL BE WRITTEN TO <logstream name> LOG STREAM FOR JOB <server name>
```

If, however, the server cannot connect to the log stream, the message is instead written to CERR, which puts it in the SYSOUT of the job output and WebSphere for z/OS continues to operate. This would be indicated by the message:

```
BB0U0025I ERRORS WILL BE WRITTEN TO CERR FOR JOB <server name>
```

## 6.6.5 Performance

Usually there are not many considerations from a performance perspective toward the WebSphere for z/OS log stream. Measurement has been taken in both configurations, with WebSphere for z/OS running with a log stream in a coupling facility and on a DASD-only log stream, and no big difference has been noticed if recent technologies has been used for DASD placement. After your initial log stream allocation, performance data is produced in a number of forms that can be useful to determine if any adjustments is necessary to your installation:

- ▶ SMF 88 data produced by the MVS System Logger
- ▶ SMF 70–78 data produced by various MVS components

There are several tools which aide in the analysis of log stream performance problems.

- ▶ IXGRPT1 formats and reports the SMF 88 data produced by the MVS System Logger
- ▶ ERBRMFPP reports on the data captured in the SMF 70–78 records

There are some fields in the SMF88 that you might want to check against this log stream: DASD SHIFT, STRUCTURE FULL, ENTRY FULL. Refer to 8.6.3, “SMF Type 88 records and IXGRPT1 program” on page 291 for a description of these fields and what to do in case one of these conditions happens in your installation.

## Logger operations

This chapter discusses normal and abnormal operations in the System Logger environment.

These topics are covered in detail:

- ▶ Display Logger command and how to interpret the results
- ▶ The LIST parameter on the IXCMIAPI utility
- ▶ How to start and stop the logger address space
- ▶ The offload monitoring tool
- ▶ HSM considerations
- ▶ How to handle the most common abnormal situations
  - Address space fails, hangs, and so forth
  - Handling structure full
  - Structure rebuild
- ▶ Tools

## 7.1 Reporting - System Logger status

There are some operator commands and utilities that are useful for gathering information about the status of System Logger, CF list structures and log streams. These facilities provide a function that reveals a great deal of information about your System Logger set up and is easy to use. Further reporting features on how to interpret the SMF88 records are explained in 8.6.3, "SMF Type 88 records and IXGRPT1 program" on page 291.

### 7.1.1 Display Logger command

The display logger command is a utility that can be used to determine the operational status of System Logger, the status of individual log streams from a local and sysplex view and determine the utilization of CF list structures. The command output is delivered via System Logger message IXG601I.

The display logger command syntax is shown in Table 7-1.

Table 7-1 Display Logger command

```
D LOGGER[,STATUS | ST ]
      [,Connection[,LSName=logstreamname[[,Jobname=mvsjobname] [,SUMM ]] ] ]
      [,Detail]
      [,Jobname=mvsjobname[[,LSNAME|,LSN=logstreamname] [,SUMM ]] ]
      [,Detail]
      [,SYSPLEX[,LSName=logstreamname]
      [,DASDONLY
      [,Logstream[,LSName=logstreamname] [,STRNAME|,STRN=structurename] ]
      [,DASDONLY
      [,STRUCTURE|,STR[,STRNAME|STRN=structurename] ]
```

See Example 7-1 for some of the more common uses of the display logger command and example output. For a detailed explanation of the entire command syntax, see *z/OS MVS System Commands, SA22-7627*, topic 4.10.26.

Example 7-1 Display Logger Command sample - LOGR address space

To check the status of the Logger address space:

```
D LOGGER,ST (status of System Logger):
IXG601I 09.07.03 LOGGER DISPLAY 058
SYSTEM LOGGER STATUS
SYSTEM SYSTEM LOGGER STATUS
-----
SYS2 ACTIVE
```

Example 7-2 Display Logger Command sample - Log streams

To check the state of a log stream and the number of system connected to the log stream - expect a vast amount of output in case there are many log streams defined in the LOGR policy:

```
D LOGGER,L or D LOGGER,LOGSTREAM
IXG601I 09.13.20 LOGGER DISPLAY 061
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM          STRUCTURE          #CONN STATUS
-----
#@$.SQ.EMHQ.LOG    I##LOGEMHQ    000000 AVAILABLE
#@$.SQ.MSGQ.LOG    I##LOGMSGQ    000000 AVAILABLE
#@$.C.#@$CCM$1.DFHLOG2  CIC_DFHLOG_001 000000 AVAILABLE
```



```

#@$.#@$CCM$1.DFHSHUN2    CIC_DFHSHUNT_001 000000 AVAILABLE
#@$.#@$CCM$2.DFHLOG2     CIC_DFHLOG_001   000000 AVAILABLE
#@$.#@$CCM$2.DFHSHUN2    CIC_DFHSHUNT_001 000000 AVAILABLE
#@$.#@$CCM$3.DFHLOG2     CIC_DFHLOG_001   000000 AVAILABLE
#@$.#@$CCM$3.DFHSHUN2    CIC_DFHSHUNT_001 000000 AVAILABLE

```

---

*Example 7-3 Display Logger Command sample - Connections*

---

To check which jobnames are connected to the log stream - this command only displays those log streams that have connectors on the system where the command has been issued:

```

D LOGGER,C or D LOGGER,CONN
IXG601I 09.18.44 LOGGER DISPLAY 068
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #@$2
LOGSTREAM          STRUCTURE          #CONN STATUS
-----
ATR.#@$#PLEX.RM.DATA  RRS_RMDATA_1      000001 IN USE
ATR.#@$#PLEX.MAIN.UR  RRS_MAINUR_1      000001 IN USE
ATR.#@$#PLEX.DELAYED.UR RRS_DELAYEDUR_1   000001 IN USE
ATR.#@$#PLEX.RESTART  RRS_RESTART_1     000001 IN USE

```

---

*Example 7-4 Display Logger Command - R/W Connections*

---

To check which jobnames are connected to the log stream and if read or write connections exists:

```

D LOGGER,C,LSN=SYSPLEX.OPERLOG,DETAIL
IXG601I 13.15.29 LOGGER DISPLAY 380
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #@$1
LOGSTREAM          STRUCTURE          #CONN STATUS
-----
SYSPLEX.OPERLOG    SYSTEM_OPERLOG    000002 LOSS OF DATA
  DUPLEXING: LOCAL BUFFERS
    JOBNAME: BARI      ASID: 003A
    R/W CONN: 000001 / 000000
    RES MGR./CONNECTED: *NONE* / NO
    IMPORT CONNECT: NO
    JOBNAME: CONSOLE  ASID: 000A
    R/W CONN: 000000 / 000001
    RES MGR./CONNECTED: *NONE* / NO
    IMPORT CONNECT: NO

```

---

*Example 7-5 Display Logger Command - Structure association*

---

To check which log streams are allocated to a particular structure:

```

D LOGGER,STR,STRN=CIC_DFHLOG_001
IXG601I 09.23.46 LOGGER DISPLAY 083
INVENTORY INFORMATION BY STRUCTURE
STRUCTURE          CONNECTED
-----
CIC_DFHLOG_001
#@$.#@$CCM$1.DFHLOG2    NO
#@$.#@$CCM$2.DFHLOG2    NO
#@$.#@$CCM$3.DFHLOG2    NO
#@$.#@$CWC2A.DFHLOG2    NO
#@$.#@$CWE2A.DFHLOG2    NO
#@$.#@$CWJ2A.DFHLOG2    NO

```

---

*Example 7-6 Display Logger Command - Connected log streams*

---

```
DISPLAY LOGGER,CONN,LSN=ATR* (display all connected log streams that start with ATR):
IXG601I 09.39.20  LOGGER DISPLAY 204
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #@$2
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----
ATR.#@$#PLEX.RM.DATA      RRS_RMDATA_1      000001  IN USE
ATR.#@$#PLEX.MAIN.UR     RRS_MAINUR_1     000001  IN USE
ATR.#@$#PLEX.DELAYED.UR  RRS_DELAYEDUR_1  000001  IN USE
ATR.#@$#PLEX.RESTART     RRS_RESTART_1    000001  IN USE
ATR.#@$#PLEX.ARCHIVE     RRS_ARCHIVE_1    000001  IN USE
```

---

## 7.1.2 LIST Option on IXCMIAPU

One of the features of the IXCMIAPU program for System Logger (DATA TYPE(LOGR)) is its reporting ability. You can specify either LIST LOGSTREAM(*lname*) or LIST STRUCTURE(*strname*) depending on the type of specific results you are looking for.

Specifying LIST STRUCTURE(*strname*) DETAIL(YES), where *strname* is the CF list structure name (wildcards are supported), generates a report listing the structure definition values, the effective average buffer size and the log streams defined to structures listed. You can obtain the effective average buffer size even without the DETAIL(YES) option but not all the other information.

Specifying LIST LOGSTREAM(*lname*) DETAIL(YES), where *lname* is the log stream name (wildcards are supported), generates a report listing all of the log streams matching the portion of the name specified. The output includes the log stream definition, names of any associated or possible orphan data sets, connection information, structure definitions for the CF-Structure based log streams. Without the DETAIL(YES) keyword, only the log stream definition are reported in the sysout.

In addition, if you need to obtain information about the Couple Data Set, their definition values and the current usage, you need to run the IXCMIAPU utility with the DATA TYPE(LOGR) REPORT(YES) keyword.

This information is valuable in determining your resource usage and planning for future migrations. The information generated from this report can also be used to rebuild your policy; we have included a sample REXX program that uses the IXCMIAPU output to generate your policy definition; see Appendix A, "Rebuilding the LOGR Policy" on page 309 for details.

For sample output of the IXCMIAPU report see Example 7-7.

*Example 7-7 IXCMIAPU Sample Report Output*

---

```
DATA TYPE(LOGR) REPORT(NO)

LIST STRUCTURE NAME(RRS_RMDATA_1) DETAIL(YES)
LIST LOGSTREAM NAME(ATR.#@$#PLEX.RM.DATA) DETAIL(YES)
IXG005I LOGR POLICY PROCESSING LINE# 4

STRUCTURE NAME(RRS_RMDATA_1) LOGSNUM(1)
      MAXBUFSIZE(1024) AVGBUFSIZE(252)
      EFFECTIVE AVERAGE BUFFER SIZE(252)

      LOGSTREAM NAME          CONNECTION
      -----
      -----
```



There are a couple of exceptions to what was mentioned earlier:

There is a short window where the report shows an orphaned data set that is actually in use. This is when a data set switch just occurred and a new offload data set has been allocated, created an entry in the catalog but not yet in the policy data set. You should realize this particular exception because the orphaned data set will be the one with the highest sequence number.

There is the possibility that some offload data sets can be marked as orphan if they are cataloged in a user catalog shared across multiple sysplexes. If your installation resembles this configuration, you should verify that orphan data sets are not in use in any of the sysplex before deleting.

The LIST option with DETAIL(YES) also identifies 'Delete Pending' data sets. These data sets are offload data sets that are eligible to be deleted by system logger but for some reason, they could not be successfully deleted. One of the most common causes is that when system logger tried to delete the data set, some application was browsing the expired data. In this case, system logger marks the data set 'Delete Pending' and will try to delete the data set at the next offload. You can manually delete these data sets if you feel comfortable that logger doesn't need this data. But we suggest that you let system logger deal with them.

### 7.1.3 Display GRS command

System Logger uses enqueues to serialize access to its resources. In case of a problem, you can verify if there is any deadlock situations by issuing the commands shown in Example 7-8.

*Example 7-8 DISPLAY GRS Command samples*

---

D GRS,C	to check if there is any deadlock situation
D GRS,LATCH,JOBNAME=IXGLOGR	to check for outstanding log streams latches
D GRS,RES=(SYSZLOGR,*)	to check for ENQ contention Major name=SYSZLOG minor= log stream name

---

In case the command returns you an outstanding enqueue, you might want to re-issue the command a few times over several minutes, and if the owner of the resource does not change, then there might be a serialization problem.

If the problem persists, refer to the 2.10.1, "Collecting documentation for problem diagnosis" on page 77 to verify which documentation is needed to debug the problem.

### 7.1.4 How to identify log streams with problems / damaged log stream

Unfortunately, there is no Display command that gives you a complete view of the log stream, checking for example that all the offload data sets described in the couple data sets actually exist on DASD and none of them have been deleted manually.

The only time that you realize you have such a problem is when you try to retrieve the data. At that time logger will account for a gap of data in the log stream and it is up to the application to recover from this situation. The exploiter symptoms, that is, messages, can also be useful to determine what to do to recover the situation.



LOG STREAM DATA SET INFO:

DATA SET NAMES IN USE: IXGLOGR.SYSPLEX.OPERLOG.<SEQ#>

Ext.	<SEQ#>	Lowest Blockid	Highest GMT	Highest Local	Status
-----	-----	-----	-----	-----	-----
*00001	A0000299	000000010D52B2CB	07/28/03 18:02:21	07/28/03 14:02:21	
	A0000300	000000010D5971D9	07/28/03 18:32:24	07/28/03 14:32:24	
	A0000301	000000010D6030F0	07/28/03 18:52:09	07/28/03 14:52:09	
	A0000302	000000010D66F0B1	07/28/03 18:53:49	07/28/03 14:53:49	
	A0000303	000000010D6DB01C	07/28/03 18:55:16	07/28/03 14:55:16	
	A0000304	000000010D746F6B	07/28/03 18:56:53	07/28/03 14:56:53	
	A0000305	000000010D7B2E9F	07/28/03 18:58:32	07/28/03 14:58:32	
	A0000306	000000010D81ED65	07/28/03 18:59:56	07/28/03 14:59:56	
	A0000307	000000010D88AD21	07/28/03 19:01:25	07/28/03 15:01:25	
	A0000308	000000010D8F6C20	07/28/03 19:02:24	07/28/03 15:02:24	
	A0000309	000000010D962BB5	07/28/03 19:03:13	07/28/03 15:03:13	
	A0000310	000000010D9CEAB2	07/28/03 19:04:03	07/28/03 15:04:03	
	A0000311	000000010DA3A9B0	07/28/03 19:04:49	07/28/03 15:04:49	
	A0000312	000000010DAA687B	07/28/03 19:05:38	07/28/03 15:05:38	
	A0000313	000000010DB127B7	07/28/03 19:06:32	07/28/03 15:06:32	
	A0000314	000000010DB7E6A1	07/28/03 19:07:14	07/28/03 15:07:14	
	A0000315	000000010DBEA614	07/28/03 19:08:02	07/28/03 15:08:02	
	A0000316	000000010DC564FA	07/28/03 19:08:34	07/28/03 15:08:34	
	A0000317	000000010DCC241C	07/28/03 19:09:07	07/28/03 15:09:07	
	A0000318	000000010DD2E30D	07/28/03 19:09:39	07/28/03 15:09:39	
	A0000319	000000010DD9A1BA	07/28/03 19:10:10	07/28/03 15:10:10	
	A0000320	000000010DE061A7	07/28/03 19:10:38	07/28/03 15:10:38	
	A0000321	000000010DE72061	07/28/03 19:11:04	07/28/03 15:11:04	
	A0000322	000000010DEDDDC	07/28/03 19:11:31	07/28/03 15:11:31	
	A0000323	000000010DF49ED8	07/28/03 19:11:59	07/28/03 15:11:59	
	A0000324	000000010DFB5EA8	07/28/03 19:12:26	07/28/03 15:12:26	
	A0000325	000000010E021E4A	07/28/03 19:12:54	07/28/03 15:12:54	
	A0000325	000000010E021E4A	07/28/03 19:12:54	07/28/03 15:12:54	
	A0000326	000000010E08DE1A	07/28/03 19:13:21	07/28/03 15:13:21	
	A0000327	000000010E0F9E16	07/28/03 19:13:48	07/28/03 15:13:48	
	A0000328	000000010E165DE6	07/28/03 19:14:15	07/28/03 15:14:15	
	A0000329	000000010E1D1D8B	07/28/03 19:14:41	07/28/03 15:14:41	
	A0000330	000000010E23DD5B	07/28/03 19:15:09	07/28/03 15:15:09	
	A0000331	000000010E2A9D54	07/28/03 19:15:36	07/28/03 15:15:36	
	A0000332	000000010E315D24	07/28/03 19:16:03	07/28/03 15:16:03	
	A0000333	000000010E381CC9	07/28/03 19:16:30	07/28/03 15:16:30	
	A0000334	000000010E3EDC99	07/28/03 19:16:57	07/28/03 15:16:57	
	A0000335	000000010E459C95	07/28/03 19:17:24	07/28/03 15:17:24	
	A0000336	000000010E4C5C62	07/28/03 19:17:51	07/28/03 15:17:51	
	A0000337	000000010E531C07	07/28/03 19:31:08	07/28/03 15:31:08	
	A0000338	000000010E59DAF4	07/28/03 19:41:23	07/28/03 15:41:23	
	A0000339	000000010E609A62	07/28/03 19:50:39	07/28/03 15:50:39	
	A0000340	000000010E6759A0	07/28/03 20:21:52	07/28/03 16:21:52	
	A0000341	000000010E6E18B3	07/28/03 20:28:27	07/28/03 16:28:27	
	A0000342	000000010E74D776	07/28/03 20:35:34	07/28/03 16:35:34	
	A0000343	000000010E7B9674	07/28/03 20:49:51	07/28/03 16:49:51	
	A0000344	000000010E8255B3	07/28/03 20:57:44	07/28/03 16:57:44	
	A0000345	000000010E891511	07/29/03 17:44:13	07/29/03 13:44:13	
	A0000346	000000010E8FD397	07/29/03 21:11:02	07/29/03 17:11:02	
	A0000347	000000010E969259	07/29/03 22:16:41	07/29/03 18:16:41	
	A0000348	000000010E9D521D	07/29/03 23:52:46	07/29/03 19:52:46	
	A0000349	000000010EA4118E	07/30/03 21:15:00	07/30/03 17:15:00	
	A0000350	000000010EAAD11D	07/30/03 21:25:38	07/30/03 17:25:38	CURRENT

NUMBER OF DATA SETS IN LOG STREAM: 52

POSSIBLE ORPHANED LOG STREAM DATA SETS:

NUMBER OF POSSIBLE ORPHANED LOG STREAM DATA SETS: 0

---

For both CF-Structure based and DASD-only log streams, System Logger marks a log stream as permanently damaged when it cannot recover log data from either DASD staging data sets or the local buffers after a system, sysplex, or coupling facility failure. Applications are notified of the damage via System Logger services and reason codes. Recovery actions are necessary only if warranted for the application.

In the case that System Logger has been unable to recover data for a log stream, its status when using the D LOGGER,L command will be "POSSIBLE LOSS OF DATA".

**Note:** You should never delete offload data sets (except orphaned offload data sets) manually. This will cause an unrecoverable loss of data.

## 7.2 Offload monitoring

As part of z/OS 1.4 (or via APAR OW51854 on earlier releases), System Logger introduced function that allows installations to monitor offloads. If an offload appears to be taking too long or if it hangs, System Logger will issue message IXG312E. There are several actions you can take to attempt to determine what may be inhibiting an offload:

1. First of all, verify that there are no outstanding WTORs.
2. Determine if there are inhibitors to offload processing by issuing such commands as:
  - D LOGGER,C,LSN=logstreamname
  - D LOGGER,L,LSN=logstreamname
  - D XCF,STRUCTURE,STRNAME=structurename
  - D GRS,CAttempt to remedy any problems noted.
3. If the problem persists, respond or react to any Allocation, Catalog, or recall messages such as IEF861I, IEF863I, or IEF458D.
4. If the problem persists, respond to message IXG312E. This can be used to stop the offload processing for the log stream named in the message and allow it to run on another system, if possible. It may also allow other work to run on the system that was attempting the original offload.
5. If message IXG115A is displayed, reply to this message only after you have attempted to remedy any delayed offloads by responding to the related IXG312E messages. As a *last* resort, if you reply "TASK=END" to an IXG115A message, System Logger will terminate all the log stream connections in the structure named in the message on this system.

Review the complete description of messages IXG311I, IXG312E, IXG114I, and IXG115I in the *z/OS MVS System Messages Volume 10 (IXC - IZP)*, SA22-7640-03, before responding to any of these messages.

Note that several messages could appear at the same time for different log stream for which offloading is taking place. It may be that only one log stream is actually having a problem, and

the others are simply waiting for this log stream to finish its allocation processing. Usually, the log stream that is causing the delay is the first log stream to be reported on.

Another way to determine which log stream is actually causing the delay is to check in the message IXG312E to see if the data set name ends with an explicit sequence number, such as "A0000001". If this is the case, it usually means that this log stream is experiencing a problem. If the data set name ends with "<SEQ#>", then it is more likely that this log stream is waiting for another log stream that is having a problem.

## 7.3 Stopping/Starting Logger Address Space

After OW53349 the Logger has been changed to automatically restart, if it was not terminated by the FORCE IXGLOGR,ARM command.

Still, if for any reason, you need to terminate the logger address space, you need to issue the FORCE IXGLOGR,ARM command and the only way to restart it is through the S IXGLOGRS procedure.

IXGLOGRS is the command processor to start the logger address space. IXGLOGRS only starts the logger address space IXGLOGR and then it immediately ends.

## 7.4 Handling Directory Full Condition

When your installation receives message IXG261E or IXG262E, it means that the system logger has detected a shortage of log data set directory extent records in the active LOGR couple data set. In this situation, the system logger may eventually fail log stream offloads if it is unable to obtain a log data set directory extent required to process the offload.

Deleting offload data sets via IDCAMS does not help to solve this situation since the LOGR couple data set keeps track of all data sets. Deleting them (via IDCAMS, for example) will not cause the LOGR couple data set to be updated, and logger will still think the data sets exists, and report missing data if an attempt is then made to access the data mapped (from the contents of the LOGR couple data set) in those data sets.

To solve this situation you can either try to understand which log stream is generating the high number of offload data sets or enlarging the extend portion of the LOGR couple data set.

To determine which log stream is using all the directory entries, you can run the IXCMIAPU utility and then take the corrective action against the log stream, as described by the above referenced messages.

If this does not solve the situation, the following is a list of actions you can take to solve the problem:

1. Run the IXCMIAPU utility with the LIST option against the log streams to verify which log streams are generating the high amount of offload data sets that are using all the inventory entries. Check also if there is any anomaly in the definition of these log streams. Very often a wrong parameter is the cause of the elevated number of offload data sets being created. For example, a small value for LS\_SIZE is very common to be found. This means very small offload data sets and if the log stream is generating a huge amount of data, this can cause many offload data sets being created using all the available directory entries.
2. Define a new LOGR CDS with a bigger DSEXTENT value that will allow new offload data sets to be allocated. This will give some time to the installation to recover.



3. The next action is to change the size of the LOGR couple data set with a larger DSEXTENT value, allowing for additional data sets to be defined for the log stream and make this new LOGR couple data set the active data set in the sysplex.
4. Fixed the problem with the log stream either deleting the log stream or updating the LOGR policy. This should fix the problem permanently.

Following is the procedure to define new LOGR couple data sets and to make them active in the sysplex. Before allocating the new data set, you can display the current allocation with the command `D XCF,COUPLE,TYPE=LOGR` or you can run the `IXCMIAPU` and look for the `DSEXTENT` field in the output display. This tells you how many extent are allocated in the current LOGR couple data set.

*Example 7-10 Current LOGR couple data set allocation*

---

```
IXC358I 16.15.44 DISPLAY XCF 521
LOGR COUPLE DATA SETS
PRIMARY DSN: SYS1.XCF.LOGR01
VOLSER: #@$#X1 DEVN: 37AC
FORMAT TOD MAXSYSTEM
12/11/2002 21:43:54 4
ADDITIONAL INFORMATION:
LOGR COUPLE DATA SET FORMAT LEVEL: HBB7705
LSR(200) LSTRR(120) DSEXTENT(10)
SMDUPLEX(1)
ALTERNATE DSN: SYS1.XCF.LOGR02
VOLSER: #@$#X2 DEVN: 37AD
FORMAT TOD MAXSYSTEM
12/11/2002 21:43:58 4
ADDITIONAL INFORMATION:
LOGR COUPLE DATA SET FORMAT LEVEL: HBB7705
LSR(200) LSTRR(120) DSEXTENT(10)
SMDUPLEX(1)
LOGR IN USE BY ALL SYSTEMS
```

---

At this point, you can use the `IXCL1DSU` utility to format a new LOGR CDS. Make sure the new CDS you format has the appropriate number of LSRs, LSTRRs, and a larger DSEXTENTS.

Once you have the new LOGR couple data set allocate, you can make this as the alternate LOGR couple data set in your installation by issuing the command `SETXCF COUPLE,ACOUPL=(new_dsname),TYPE=LOGR`.

If the addition of this couple data set is successful, then you can proceed and issue the `SETXCF COUPLE,TYPE=LOGR,PSWITCH` to switch control from the current primary to the new allocated one. Remember to allocate the new alternate since in this moment the installation is running without an alternate LOGR couple data set.

## 7.5 Handling of Couple Data Sets

LOGR couple data sets contains the current LOGR policy and also the inventory of the offloads data sets. This means that system logger uses the inventory to understand on which data set the data for a particular log stream is located once that they are offloaded from the interim storage.

For this reason it is important that the current LOGR couple data set are preserved across IPL. For this reason:

- ▶ Every time you switch to a new LOGR couple data set, you should remember to update the COUPLExx parmlib member to point to the new data sets to avoid inconsistency when you will IPL later on.
- ▶ Every time that you perform an IPL without having updated the parmlib member, you will receive message IXC287I to describe the discrepancy between the parmlib member and the last used data set. In this case, you should reply 'U' to message IXC289D to continue the IPL with the last used couple data sets and update the parmlib immediately after IPL completes.

## 7.6 HSM considerations

There are two situations where HSM recall might impact your application performance:

- ▶ When your application is connecting to the log stream whose data could not be previously offloaded. For this reason, current data are still in the interim storage and during connect process system logger attempts to offload the data which results in the allocation of the most recent offload data set.
- ▶ After the application connects to a log stream and performs multiple IXGWRITE operations that result in hitting the highoffload threshold. System logger initiates an offload which result in the allocation of the most recent offload data set.

In these cases, if your installation is aggressively migrating offload data sets after a short period of time, there might be the possibility that the current offload data set might have been migrated while the application was not running. In these cases, the HSM recall might introduce a delay in your application.

To avoid undesired behavior, you can define whether offload processing is to recall the current offload data.

On the DEFINE LOGSTREAM keyword, you can specify the OFFLOADRECALL parameter to control this process. The values of the keywords are:

<b>YES</b>	Offload processing should recall the current offload data set.
<b>NO</b>	Offload processing should skip recalling the current offload data set and allocate a new one.

**Note:** When you chose not to recall the current offload data set, you should be aware that this might cause some wasted space on DASD once it is recalled since it was not a totally filled up data set. Care should be taken when using this option to size the data sets appropriately.

You can update this function at any time during normal operations. This keyword can be updated even when the log stream is actively connected. In this case the change will be marked as 'pending' and takes effect on the subsequent first connection to the log stream in the sysplex:

- ▶ For a structure-based log stream, the change will also take effect during the next structure rebuild
- ▶ For a DASD-only log stream, the change also will take effect upon the next offload data

## 7.7 Deletion of a log stream with valid data in interim storage

There are situations where an installation would like to delete a log stream that still has valid data in the interim storage that logger has trouble offloading, for example when there is no more space on the offload volume(s).

In such a situation, logger will create an internal connection to the log stream to be able to perform the offload.

There are two ways to delete such a log stream:

- ▶ You need to establish a connection to the log stream that will cause logger to attempt recovery for that log stream. You can generate a connection to a given log stream by restarting the application or using utilities provided by the application, or simply using a job that will try to force a connection to the log stream as in the Example 7-11.

*Example 7-11 Sample job to force a connection to log stream by using SUBSYS on DD*

---

```
//IEBGENER JOB ,CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),NOTIFY=&SYSUID.  
//COPY EXEC PGM=IEBGENER  
//SYSIN DD DUMMY  
//SYSPRINT DD SYSOUT=*  
//SYSUT1 DD DSN=logstream_name,SUBSYS=(LOGR,IXGSEXIT),  
// DISP=SHR,DCB=(RECFM=VB,BLKSIZE=32760)  
//SYSUT2 DD SYSOUT=*
```

---

- ▶ The straight way to delete the log stream without performing the offload of the data from the interim storage is to run the DELETE LOGSTREAM on the system where the internal connection exists. In this case, logger will be able to remove the internal connection and perform the deletion of the log stream without completing the offload request.

Here is a list of the recommended steps to perform the deletion of the log stream in such a situation:

1. Issue the D LOGGER,L,LSN=LOGGER.TEST where LOGGER.TEST is the log stream that is having offload problem with associated message IXG301I RC8 RSN805. This should give you an idea of how many connections exist against the log stream and from which system.
2. Issue the D LOGGER,C,LSN=LOGGER.TEST to verify the connections against the log stream. If the reply message IXG601I does not contain any information, this means that the connection described in the previous command is an internal connection.
3. At this point, if you want to delete the log stream, you need to run the DELETE LOGSTREAM NAME(LOGGER.TEST) command on the system that owns the internal connection. You find the system as part of the reply of the D LOGGER,L,LSN=LOGGER.TEST command.

*Example 7-12 Deletion of the log stream with valid data in interim storage*

---

```
IXG301I SYSTEM LOGGER FAILED TO OFFLOAD DATA FOR LOG STREAM 742  
LOGGER.TEST IN STRUCTURE LOG_TEST_001. RETURN CODE: 00000008 REASON  
CODE: 00000805 DIAG1: 00000004 DIAG2: 4714041D DIAG3: 0107001B  
DIAG4: 00000000
```

```
D LOGGER,L,LSN=LOGGER.TEST  
IXG601I 14.22.03 LOGGER DISPLAY 744  
INVENTORY INFORMATION BY LOGSTREAM  
LOGSTREAM STRUCTURE #CONN STATUS  
-----  
LOGGER.TEST LOG_TEST_001 000001 IN USE
```

```

SYSNAME: #@$3
DUPLEXING: LOCAL BUFFERS

D LOGGER,C,LSN=LOGGER.TEST
IXG601I 14.24.41  LOGGER DISPLAY 746
CONNECTION INFORMATION BY LOGSTREAM FOR SYSTEM #@$3
LOGSTREAM          STRUCTURE          #CONN STATUS
-----          -
NO MATCHING INFORMATION FOUND.

```

---

At this point, to delete the log stream with still valid data in the interim storage, you need to submit the following JCL on system #@\$3 that is the one owing the connection (Example 7-13).

*Example 7-13 Sample job to delete the log stream*

---

```

//POLLOGR JOB MSGLEVEL=(1,1),MSGCLASS=H,CLASS=A
//*
//*****
/*JOBPARM SYSAFF=#@$3
//STEP20 EXEC PGM=IXCMIAPU,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *

DATA TYPE(LOGR) REPORT(NO)

DELETE LOGSTREAM NAME(LOGGER.TEST)
LIST LOGSTREAM NAME(LOGGER.TEST) DETAIL(YES)

```

---

## 7.8 Structure rebuilding

This section applies to coupling facility log streams only.

The following are the main reasons to initiate the rebuilding of a structure that contains log stream data:

- ▶ Operator request because of the need of moving allocated storage from one coupling facility to another one
- ▶ Reaction to a failure
- ▶ Implement a new CFRM policy

### ***Operator request***

An operator can initiate the rebuild of the structure because of the need to either change the configuration of the coupling facility, put the coupling facility offline due to maintenance request or altering the size of the coupling facility structure. The rebuild operation can happen dynamically while application are connected to the log stream.

While the rebuild is in progress, system logger rejects any system logger service requests against the log stream. Since this is only a temporary condition, most exploiters simply report the failed attempt and re-drive it.

In order to allow the moving of the structure from one coupling facility to another you need to specify the potential alternate coupling facility on your preference list as described in Example 7-14 and if you are planning to increase the size of you structure, you need to specify INITSIZE and SIZE in order to define the initial storage allocate and the potential growth.

*Example 7-14 CFRM policy Sample*

---

```
STRUCTURENAME(STRUCT1) SIZE(7000)
INITSIZE(4000)
MINSIZE(2000)
FULLTHRESHOLD(0)
ALLOWAUTOALT(NO)
PREFLIST(CF01,CF02,CF03)
```

---

Your installation can decide to have the system automatically altering a structure when it reaches an installation-defined or defaulted-to percent full threshold. In order to use this function, you need to define the ALLOWAUTOALT parameter to YES in the CFRM policy.

If you specify the auto alter function, when an application connects to a coupling facility log stream using the IXGCONN macro, System Logger issues the following request to connect to the coupling facility list structure:

```
IXLCONN ..... ALLOWALTER(YES)
```

When these two tasks happen, XES can then automatically alter the size and ratio attributes of the structure as needed.

Although possible, specifying ALLOWAUTOALT is not recommended because system logger does it own deletion of the coupling facility space, and at times, it might look like logger is not using its space. This behavior might lead XCF to wrong calculation.

System Logger manages the usable space within its own structure by allowing each log stream to receive 1/n of the portion of the number of active connections. System Logger does offloading of log data from the coupling facility to DASD offload data sets when their specified thresholds are met. So it is very possible that logger structures will appear to be only half full and logger structures are also good candidates for XES to take its space for other constrained structures.

Because of these facts you should consider allowing Auto Alter processing for other structures than logger structures first. If you specify ALLOWAUTOALT(YES) in the CFRM policy, you also specify the parameter MINSIZE(n) in the same policy. The value of n should be chosen in a way that enables the logger to handle the expected number of concurrent active log stream connections.

These commands issue whether to initiate a structure rebuild or an alteration:

```
SETXCF START,REBUILD,STRNAME=structure_name
SETXCF START,ALTER,STRNAME=structure_name,SIZE=5000
```

*Example 7-15 Rebuild process sample*

---

```
SETXCF START,REBUILD,STRNAME=RRS_MAINUR_1
```

```
IXC521I REBUILD FOR STRUCTURE RRS_MAINUR_1 HAS BEEN STARTED
IXC367I THE SETXCF START REBUILD REQUEST FOR STRUCTURE RRS_MAINUR_1 WAS ACCEPTED.
IXG117I STRUCTURE REBUILD STARTED FOR STRUCTURE RRS_MAINUR_1
      1 OF 1 LOGSTREAMS CONNECTED TO STRUCTURE:
      ATR.#@$#PLEX.MAIN.UR
IXC526I STRUCTURE RRS_MAINUR_1 IS REBUILDING FROM
```

```

COUPLING FACILITY FACIL06 TO COUPLING FACILITY FACIL05.
REBUILD START REASON: OPERATOR INITIATED
INFO108: 00000064 00000064.
IXL014I IXLCONN REBUILD REQUEST FOR STRUCTURE RRS_MAINUR_1
WAS SUCCESSFUL. JOBNAME: IXGLOGR ASID: 0014
CONNECTOR NAME: IXGLOGR_#@$2 CFNAME: FACIL05
IXL015I REBUILD NEW STRUCTURE ALLOCATION INFORMATION FOR 692
STRUCTURE RRS_MAINUR_1, CONNECTOR NAME IXGLOGR_#@$2
CFNAME ALLOCATION STATUS/FAILURE REASON
-----
FACIL01 INSUFFICIENT CONNECTIVITY 00000000
FACIL02 INSUFFICIENT CONNECTIVITY 00000000
FACIL03 INSUFFICIENT CONNECTIVITY 00000000
FACIL04 INSUFFICIENT CONNECTIVITY 00000000
FACIL05 STRUCTURE ALLOCATED
FACIL06 PREFERRED CF ALREADY SELECTED
IXL014I IXLCONN REBUILD REQUEST FOR STRUCTURE RRS_MAINUR_1
WAS SUCCESSFUL. JOBNAME: IXGLOGR ASID: 0014
CONNECTOR NAME: IXGLOGR_#@$1 CFNAME: FACIL05
IXL014I IXLCONN REBUILD REQUEST FOR STRUCTURE RRS_MAINUR_1
WAS SUCCESSFUL. JOBNAME: IXGLOGR ASID: 0014
CONNECTOR NAME: IXGLOGR_#@$3 CFNAME: FACIL05
ATR218I TAKEOVER PROCESS HAS FAILED DUE TO
INACCESSIBLE LOG DATA IN LOGSTREAM ATR.#@$#PLEX.MAIN.UR.
IXC521I REBUILD FOR STRUCTURE RRS_MAINUR_1 861
HAS BEEN COMPLETED
IXG111I STRUCTURE REBUILD FOR STRUCTURE RRS_MAINUR_1 IS COMPLETE.
LOGSTREAM DATA DEFINED TO THIS STRUCTURE IS AVAILABLE
IXC579I PENDING DEALLOCATION FOR STRUCTURE RRS_MAINUR_1 IN
COUPLING FACILITY 002066.IBM.02.000000011CE3
PARTITION: 05 CPCID: 00
HAS BEEN COMPLETED.
PHYSICAL STRUCTURE VERSION: B9C318D4 C9523601
INFO116: 13088068 01 6A00 0000000F
TRACE THREAD: 00176EE9.
IXG111I STRUCTURE REBUILD FOR STRUCTURE RRS_MAINUR_1 IS COMPLETE.
LOGSTREAM DATA DEFINED TO THIS STRUCTURE IS AVAILABLE
IXG111I STRUCTURE REBUILD FOR STRUCTURE RRS_MAINUR_1 IS COMPLETE.
LOGSTREAM DATA DEFINED TO THIS STRUCTURE IS AVAILABLE

```

---

## Reaction to failure

The following coupling facility problems can occur, resulting in rebuild processing for the structure:

- ▶ Damage to or failure of the coupling facility structure.
- ▶ Loss of connectivity to a coupling facility.
- ▶ A coupling facility becomes volatile.

No matter which is the source of the problem, coupling facility or connectivity, system logger initiates a rebuild to move the structure to another coupling facility to avoid loss of data. The rebuild is initiate independently of the REBUILDPERCENT specified on the policy definition for this structure.

### ***Damage to or Failure of the Coupling Facility Structure***

If a structure was in duplex-mode and a coupling facility failure or structure damage occurs to only one instance of the structure, then XES will automatically switch from duplex-mode to simplex-mode. Logger will only be notified of the mode switchchange and will not be aware of any coupling facility failure or damage to the structure.

If DUPLEX has been defined as ENABLED the duplexing is re-established by XES; if DUPLEX is defined as ALLOWED, then operator must issue the command to restart duplexing.

If the structure was in simplex-mode and the failure or damage occurs to the only instance of the structure, then all systems connected to the coupling facility structure detect the failure. The first system whose system logger component detects the failure initiates the structure rebuild process. The structure rebuild process results in the recovery of one or more of the affected coupling facility structure's log streams. All the systems in the sysplex that are connected to the list structure participate in the process of rebuilding the log streams in a new coupling facility list structure.

While the rebuild is in progress, system logger rejects any system logger service requests against the log stream. The status will be one of the following:

- ▶ The structure rebuild has completed successfully, the coupling facility structure and associated log streams are available, and system logger requests will be accepted.
- ▶ The structure rebuild was unsuccessful and connection to the structure is not possible because the structure is in a failed state. Log data still resides in staging data sets if they are used to duplex the log data for the log stream. If staging data sets were not used, the data persists in the local storage buffers on each system. System Logger dynamically allocates staging data sets to prevent a possible loss of data.

### ***Loss of Connectivity to the Coupling Facility Structure***

If a structure was in duplex-mode and a loss of connectivity occurs to only one instance of the structure (due to a hardware link failure), then XES automatically switches from duplex-mode to simplex-mode. Logger is only notified of the mode switch change and is not aware of any loss of connectivity to the structure.

If DUPLEX has been defined as ENABLED the duplexing is re-established by XES; if DUPLEX is defined as ALLOWED, then operator must issue the command to restart duplexing.

For a simplex-mode structure, Logger detects the loss of connectivity to the single instance of the structure. Then the system that lost connectivity may initiate a rebuild for the structure.

System logger rejects logger service requests issued during the rebuild process. If XES cannot allocate a new structure instance in a coupling facility that can be connected to all the affected systems, system logger does one of the following, depending on whether the system or systems that cannot connect to the new coupling facility structure were using staging data sets:

- ▶ If the system was using staging data sets, the rebuild process continues and the coupling facility log data for the system is recovered from the staging data sets.
- ▶ If the system was not using staging data sets, the rebuild process is stopped. The systems go back to using the source structure. The application on the system that loses connectivity to the coupling facility will lose connectivity to the log stream and they can either fail or wait depending on their recovery process.

The systems that do not have connectivity to the old coupling facility structure issue an ENF 48 event indicating that they do not have connectivity to the log stream. The systems that can connect to the source structure issue an ENF 48 event indicating that the log stream is available to that system and can resume use of the log stream.

The installation should either update the CFRM active policy to make the new coupling facility structure available to all the systems or else fix the hardware link problem and then have the

operator initiate a rebuild for the structure so that all the original systems will have connectivity.

## 7.9 Tools

The following are tools that are useful to control the system logger environment:

### 7.9.1 DUMP command parmlib member

IEADMCxx enables you to supply DUMP command parameters through a parmlib member and to specify the collection of dump data without having to remember and identify all the systems, address spaces and data spaces involved.

The following is a recommended sample for the DUMP command to identify all the required documentation you need to collect to debug a system logger problem.

*Example 7-16 IEADMC parmlib member sample*

---

```
TITLE =(DUMP OF LOGGER AND RELATED DATA),
JOBNAME=(IXGLOGR,XCFAS,ALLOCAS,SMS*,CATALOG,GRS,DFHSM),
DSPNAME=('XCFAS'.*, 'IXGLOGR'.*, 'GRS'.*, 'SMS*'.*),
SDATA =(COUPLE,ALLNUC,LPA,LSQA,PSA,SWA,RGN,SQA,TRT,CSA,GRSQ,
        XESDATA,SUM),
STRLIST=(STRNAME=&STRNAME.&STRNAME2., LOCKENTRIES,ACC=NOLIM,
        (LISTNUM=ALL,ENTRYDATA=SERIALIZE,ADJUNCT=CAPTURE)),
REMOTE =(SYSLIST=('IXGLOGR', 'XCFAS', 'ALLOCAS', 'SMS*',
        'CATALOG', 'GRS', 'DFHSM'), DSPNAME, SDATA)
```

---

You can use the IEADMC parmlib member facility, prepare the member in parmlib and at the time of the error, simply enter the DUMP TITLE=(LOGGER PROBLEM),PARMLIB=xx command or you can use the standard DUMP command with all the above mentioned parameters.

For further information about how to set up the IEADMC parmlib member, refer to *z/OS MVS Initialization and Tuning Reference, SA22-7592*.

### 7.9.2 SMF88

System logger report data through SMF88. To collect SMF88 records, you need to request them through the parmlib member SMFPRM in the SYS and SUBSYS keywords either specifically or as part of a range of SMF records. If you want to change the recording after IPL, remember to issue the SET SMF=xx (where xx is the suffix of the SMFPRM member) command to activate the parmlib changes. The following is an example of an SMFPRM member to collect the logger SMF88:

*Example 7-17 SMFPRM member*

---

```
NOBUFFS(MSG) /* DEFAULT TO MESSAGE
SYS(TYPE(30,70:79,88,89,100,101,110),
EXITS(IEFU83,IEFU84,IEFU85,IEFACTRT,
      IEFUJV,IEFUSI,IEFUJP,IEFUSO,IEFUTL,IEFUAV),
      INTERVAL(SMF,SYNC),NODETAIL)

SUBSYS(STC,EXITS(IEFU29,IEFU83,IEFU84,IEFU85,IEFUJP,IEFUSO,
      IEFACRT),
      INTERVAL(SMF,SYNC),
```



### 7.9.3 IXGRPT1

IXGRPT1 is available in SYS1.SAMPLIB. This program can help you analyze system logger SMF88 data for the systems in a sysplex. IXGRPT1 provides the following:

- ▶ System logger interim storage related I/O activity
- ▶ Nearness of the STRUCTURE FULL condition
- ▶ Selected capacity planning information

The input to IXGRPT1 should be sorted by timestamp and log stream name. For sorting purposes, analysis programs should use timestamp field SMF88LTD (note that this field is in GMT format), created when the ENF signal was issued, rather than fields SMF88TME and SMF88DTE, which indicate when a particular record was written. If IXGRPT1 detects a sorting error in the input, an error message is produced and the program ends.

When you use the IXGRPT1 program, make sure to include type 88 subtype 1 records and indicate whether or not you wish to include DASD-only log stream information in this report or coupling facility data only.

Follow the instructions in the prolog of IXGRPT1 to run the utility or refer to JCL sample IXGRPT1J to run the utility.

The following is a list of information that can be obtained through the IXGRPT1 tool to document the interim storage I/O activity:

- ▶ Number of bytes written by users via IXGWRITE during the interval (SMF88LWB)
- ▶ Number of bytes written to interim storage during the interval (SMF88SWB)
- ▶ Number of bytes written to DASD during the interval (SMF88LDB)
- ▶ Number of bytes deleted from interim storage during interval without having been written to the log data set (SMF88SIB)
- ▶ Number of deletes from interim storage during interval without having been written to the log data set (SMF88SII)
- ▶ Number of bytes written to the DASD log data set and then deleted from interim storage during the interval (SMF88SAB)
- ▶ Number of deletes from interim storage during interval written to the DASD log data set and then deleted (SMF88SAI)
- ▶ Number of times the log stream was offloaded during interval (SMF88EO)
- ▶ Number of times a request was made by system logger to write log stream data to DASD during the expiring SMF interval (SMF88LIO)
- ▶ Number of times system logger had to suspend before writing log stream data to DASD because a previously-initiated write to DASD had not yet completed during the expiring SMF interval (SMF99LIS)

IXGRPT1 gives you also information about the nearness of STRUCTURE FULL condition for coupling facility log streams as follows. By analyzing the following fields, you can have an idea if the storage for the coupling facility structure needs to be tuned. These informations apply only to coupling facility log streams; for DASD-only log streams, these fields will be zeros.

- ▶ Number of IXGWRITE invocations of completion type-1 (structure fullness in “normal” range)

- ▶ Number of IXGWRITE invocations of completion type-2 (structure fullness in “warning” range)
- ▶ Number of IXGWRITE invocations of completion type-3 (structure fullness in “critical” range)
- ▶ Number of times all log stream in structure offloaded during interval
- ▶ Number of times the structure full condition was actually reached

#### **7.9.4 LOGR Couple Data Set Audit Tool**

At the moment, there are no tool to audit the content of the LOGR CDS and compare it against the catalog to validate the existence of the offload data sets and highlight any discrepancies. If an installation has this need, something has to be manually built comparing the output of the IXCMIAPU LIST command and the LISTCAT report.



## System Logger performance and tuning

This chapter provides generic information to help you tune the use of any log stream, regardless of the location of the log stream or the specifics of the exploiter that uses that log stream. Specifically, it discusses:

- ▶ The sizing of log streams, both the interim storage portion and the offload data sets
- ▶ The impact of log stream data duplexing
- ▶ How to set the high and low offload thresholds
- ▶ Other recommendations that apply to all log streams
- ▶ How to format and interpret the performance data that is available relating to System Logger

## 8.1 Introduction

There are many exploiters of System Logger. As discussed in Chapter 1, “Introduction to System Logger” on page 1, some of these predominately only write to System Logger, with very infrequent reading back of that information (the *funnel-type* exploiters). And others use the log stream data much more actively (the *active* exploiters). In addition, some exploiters write large volumes of data to System Logger, while others only write infrequently. However, because the process of writing data to System Logger is a synchronous one, it is important to all exploiters that System Logger provides good performance.

In addition, the two types of users have different requirements of System Logger. The funnel-type exploiter is mainly concerned that the log stream never fills up, and that offloads happen as efficiently as possible. The active exploiters ideally want all their data to always reside in the interim storage, with few, if any, log records being moved out to offload data sets before they are deleted.

Beyond the considerations that are unique to one or other type of exploiter, there are some general recommendations that apply to both. We discuss these in this chapter as well.

And finally, if you are to undertake an exercise to investigate and possibly tune the performance of System Logger, you must be familiar with the information provided in the SMF Type 88 records, and how best to access that information.

The following sections provide recommendations to help you set up an efficient and highly-available System Logger environment. Note, however, that some of these recommendations may be different for specific exploiters; in that case, the recommendations provided in each exploiter chapter (Chapter 3 through Chapter 6) will override the recommendations we make here.

## 8.2 Estimating log stream sizes

Log streams consist of two portions—the part kept in interim storage (in the CF or a staging data set), and the part kept in offload data sets. For the most efficient performance and use of space, you need to size the two of these individually.

### 8.2.1 Sizing offload data sets

We start with the easier one—the offload data sets. The sizing of these is relatively simple. The total of all the offload data sets for a log stream should be enough to hold the maximum amount of data you plan to keep in the log stream. For simplicity, assume that you will have a maximum of 168 offload data sets per log stream.

For an active exploiter, you would generally not expect to see more than one or two offload data sets, and each would typically not contain many log records. Therefore, the LS\_SIZE for these offload data sets would typically be in the range of 10000 to 20000 (40 MB to 80 MB).

For the funnel-type exploiters, the first thing we need to know is how long you will keep the data in the log stream. This could be as small as a number of hours, or as long as many days. The duration will vary from one exploiter to another, and from one installation to another.

The other information you need to know is how much data is moved to the offload data sets in a busy hour or a busy day. This information can be obtained from the IXGRPT1 report, as described in “BYT Deleted interim ST w/DASD”. Refer to 8.6.4, “IXGRPT1 Field Summary and IXGSMF88 Cross Reference” on page 293 for a description of the field. Remember that

each record in the IXGRPT1 report only reflects the data for one SMF interval, so you will need to total all the intervals for the duration you are interested in.

Once you know roughly how much data you wish to keep in the log stream, you need to decide on the size of each offload data set. You have two conflicting requirements here:

- ▶ You don't want to make the offload data sets so large that System Logger has difficulty finding that much free space on a volume. Also, as discussed in "Deleting log data" on page 64, old offload data sets may only get deleted when System Logger allocates a new offload data set, so if the offload data sets are many times larger than necessary, you could end up just wasting DASD space because new offload data sets rarely get allocated and old ones could be kept on DASD for far longer than necessary.
- ▶ On the other hand, having to allocate a new offload data set slows down the offload process. And every time you go through allocation, you face the risk that you won't be able to obtain the required space, or of contention problems within the allocation process, both of which will stop the offload from proceeding. So, you don't want to be allocating offload data sets too frequently either.

On balance then, try to select a size that will hold at least 10 offload's worth of log data, while staying within an amount of space that is easily obtainable in your installation. Also, try to select a size that does not hold more than one day's worth of data.

## 8.2.2 Sizing interim storage

Whereas the sizing of offload data sets is driven largely by the need to ensure that you have enough space for all the log data, the size of the interim storage is driven by the need to deliver the required performance and also by the need to ensure that interim storage does not fill up on you.

### Sizing CF-Structure log streams

The appropriate sizing of the interim storage part of CF log streams is vital in ensuring that System Logger provides the performance and availability you require. However, the method used to calculate the size differs depending on whether the log stream is a funnel-type or active log stream.

#### *Sizing interim storage for funnel-type CF log streams*

For funnel-type exploiters, the CF part of the log stream must be large enough that you don't offload too frequently, while at the same time ensuring that you can hold all the log data that will be generated at peak times within the desired offload rate.

Ideally, offload processing should not be invoked more frequently than once every 30-60 seconds. On the other hand, if you run for a whole SMF interval at peak times and have no offloads for a log stream, the interim storage is probably larger than necessary.

To estimate the amount of space required in the CF part of the log stream to hold this much data, you need to use an IXGRPT1 report. Example 8-1 contains sample JCL to run an IXGRPT1 report for just the log stream you are interested in. You should run this against one day's worth of SMF 88 data for a representative busy day.

*Example 8-1 Sample job to report on a single log stream*

```
//IXGRPT1L JOB 'Sample 88 pgm','Stephen Anania',
// MSGLEVEL=(1,1),CLASS=A,NOTIFY=&SYSUID.,MSGCLASS=H
//* *****
//* This jobs formats the SMF88s for a single log stream. *
//*
```

```

/* Unfortunately, the SORT SYSIN does not support symbols, so you *
/* must use ISPF EDIT CHANGE command to set the desired logstream. *
/* Update 'logstream_name' to the name of the logstream you want. *
/* *
/* *****
// SET SMFIN1=#@OPR17.SMFDATA.G0003V00
// SET LIBPRFX=CEE
// SET RPTOBJ=SYS1.SAMPLIB(IXGRPT1L)
//EXTRACT EXEC PGM=IFASMFDP
//SYSUDUMP DD DUMMY
//SYSPRINT DD SYSOUT=*
//SMFG00 DD DISP=SHR,DSN=&SMFIN1
//SMFALL DD DSN=&&SMF88S,UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(CYL,(5,5))
//SYSIN DD *
INDD(SMFG00,OPTIONS(DUMP))
OUTDD(SMFALL,TYPE(88))
/*
/* SORT EACH SMF INPUT DSN BY TIMESTAMP AND LOGSTREAM NAME
//SORT1 EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=&SMF88S,DISP=(OLD,KEEP)
//SORTOUT DD DSN=&&TEMP1,DISP=(NEW,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1))
//SYSIN DD *
OPTION VLSHRT
SORT FIELDS=(133,8,BI,A,
105,26,CH,A)
INCLUDE COND=(23,2,BI,EQ,X'0001',AND,
105,26,CH,EQ,C'logstream_name')
/*
//ASMC EXEC PGM=ASMA90,REGION=0M,PARM='OBJECT,NODECK' *
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR * LIBRARY DSN
//SYSLIN DD DSN=&&OBJ(IXGR1A),SPACE=(TRK,(10,30,10),,ROUND),
// UNIT=3390,DISP=(MOD,PASS),
// DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB) * ASM OBJ
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN DD DSN=SYS1.SAMPLIB(IXGR1A),DISP=SHR * ASM SOURCE
/*
/* *****
/* LINK/EDIT, AND EXECUTE OBJECT SAMPLIB MEMBER USING LE LIBRARIES.
/* *****
//LKED EXEC PGM=IEWL,REGION=1024K
//SYSLIB DD DSN=&LIBPRFX..SCEELKED,DISP=SHR * CATALOGUED PROC
//SYSPRINT DD SYSOUT=*
//SYSLIN DD DSN=&RPTOBJ.,DISP=SHR *
// DD DDNAME=SYSIN
//SYSLMOD DD DSN=&&LOADMOD(GO),DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(1024,(50,20,1)) * LOAD MODULE
//SYSUT1 DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),
// DCB=BLKSIZE=1024
//SYSIN DD DSN=*.ASMC.SYSLIN,DISP=(MOD,PASS) * ASM OBJ (IN)
//GO EXEC PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED),
// REGION=2048K
//STEPLIB DD DSN=&LIBPRFX..SCEERUN,DISP=SHR * CATALOGUED PROC
//SYSPRINT DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SMFDATA DD DSN=*.SORT1.SORTOUT,DISP=(MOD,PASS)

```

In the resulting report, select the interval that has the highest “BYT Written to interim storage” value. Divide this value by the number of minutes in the interval to get the per-minute write rate. Increase this value by 50% to allow for unexpected spikes in activity and control data. The resulting value indicates how much space there should be in the interim storage for this log stream between the HIGHOFFLOAD and LOWOFFLOAD thresholds<sup>1</sup>. Because the LOWOFFLOAD threshold for funnel-type log streams should always be 0%, we can ignore this value in our calculations.

For example, assume the SMF88SWB for the busiest interval is 31493284 and the SMF interval is 15 minutes. Dividing the 31493284 by 15 minutes gives a write rate of roughly 2 MB a minute. Increasing by 50% brings this up to 3 MB a minute. If the HIGHOFFLOAD threshold for this log stream is 70%, the CF structure should be sized so that the interim storage for this log stream is roughly 4.25 MB (3 MB / 70%). This would result in an offload approximately every minute during peak times. We will come back to the values to use for HIGHOFFLOAD and LOWOFFLOAD in 8.4.1, “High and low offload thresholds for funnel-type log streams” on page 284.

When calculating the structure size, you need to remember that the space in the structure is divided equally between all the connected log streams in that structure. You also need to allow something for the control space in the structure. So, if the structure normally contains 10 connected log streams, you would size the structure to be 4.25 MB x 10 plus 4 MB (based on CFLevel 12) for control space, giving a structure size of 46.5 MB. This assumes that all the log streams have similar write rates—if some of the log streams sharing the same structure have higher write rates, you should use the write rate of those log streams to determine the structure size.

### ***Sizing interim storage for active CF log streams***

For active exploiters, you must use a different methodology to calculate the amount of space in the CF part of the log stream. This is because you ideally want all log data to be deleted before it is offloaded to DASD. To achieve this, you need to work with both the amount of space available in the interim storage of this log stream, and also the HIGHOFFLOAD and LOWOFFLOAD thresholds.

Once again, we must use information from the IXGRPT1 report. There are two indicators that you must watch for:

- ▶ The frequency of offload processing. At peak times, there should not be more than one offload every 30 seconds, and *at a minimum* there should be *at least* one offload per SMF interval.
- ▶ The amount of data that is moved to an offload data set. This information is kept in field SMF88LDB in the SMF Type 88 record, and is shown in the field entitled “BYT Written to DASD” in the IXGRPT1 report. For an active log stream, this should always be zero.

To size interim storage for an active log stream, you need to arrive at a size, and LOWOFFLOAD and HIGHOFFLOAD values, that will result in about one offload per minute at peak times and 0 bytes being moved to offload data sets.

To do this, we need some more information from the IXGRPT1 report. We need to know the rate at which data is being written to the log stream—we can get that from the “BYT Written to interim storage” field. Remember to divide this value by the number of minutes in the SMF interval.

<sup>1</sup> To be completely accurate, we should take the CF element size into account when calculating the interim storage requirement. However the element size is sufficiently small that it will not have a significant impact.

We also need to know how often data is being logically deleted from the log stream (using IXGDELETE). If there are no IXGDELETs issued between two consecutive offloads, System Logger has no choice but to move the oldest data from interim storage to an offload data set. For some exploiters, like CICS, you can affect how often an IXGDELETE is done by adjusting the Activity Keypoint Frequency. However for exploiters where you cannot control this, you need to tune the frequency of offloads by adjusting the interim storage size—the larger the interim storage, the less frequently will an offload be initiated. You want to make sure there is at least one IXGDELETE per offload. You can find the number of IXGDELETs in the interval by summing the “# Deletes w/o DASD write” and “# Deletes w/ write fields”. Divide this value by the number of minutes in the SMF interval to determine the number of IXGDELETs per minute.

So, now we know how many bytes are being written per minute, and how many IXGDELETs are being issued per minute. If you have more than one IXGDELETE per minute, you should aim to have one offload per minute. If you only have one IXGDELETE every x minutes, you should aim to have one offload every x minutes.

To calculate the interim storage size, multiply the write rate per minute by the number of minutes between offloads. Increase this value by 50% to allow for control information and workload spikes. The result is the amount of space in the interim storage between the LOWOFFLOAD and HIGHOFFLOAD thresholds. To get the total interim storage space for this log stream, divide this amount by the difference between the HIGHOFFLOAD and LOWOFFLOAD percents. So, if the amount of data is 3 MB, and the HIGHOFFLOAD is set to 80 and the LOWOFFLOAD is set to 60, you should have  $3 \text{ MB} / (80\% - 60\%)$ , or 15 MB of interim storage for this log stream. We will come back to the values to use for HIGHOFFLOAD and LOWOFFLOAD in 8.4.2, “High and low offload thresholds for active log streams” on page 284.

When calculating the structure size, you need to remember that the space in the structure is divided equally between all the connected log streams in that structure. You also need to allow something for the control space in the structure. So, if the structure normally contains 5 connected log streams, you would size the structure to be 15 MB x 5 plus 4 MB (based on CFLevel 12) for control space, giving a structure size of 79 MB. This assumes that all the log streams have similar write rates—if some of the log streams sharing the same structure have higher write rates, you should use the write rate of those log streams to determine the structure size.

### **Sizing DASD-only log streams**

The appropriate sizing of the interim storage part of DASD-only log streams is important in ensuring that System Logger provides the performance and availability you require. However, as for CF log streams, the method used to calculate the size differs depending on whether the log stream is a funnel-type or active log stream.

#### ***Sizing interim storage for funnel-type DASD-only log streams***

For funnel-type exploiters, the staging data set part of the log stream must be large enough that you don't offload too frequently, while at the same time ensuring that you can hold all the log data that will be generated at peak times within the desired offload rate.

Ideally, offload processing should not be invoked more frequently than once every 30-60 seconds. On the other hand, if you run for a whole SMF interval at peak times and have no offloads for a log stream, the interim storage (staging data set) is probably larger than necessary.

To estimate the amount of space required in the staging data set part of the log stream to hold this much data, you need to use an IXGRPT1 report. Example 8-2 on page 279 contains



sample JCL to run an IXGRPT1 report for just the logstream you are interested in. Again, you should run this against one day's worth of SMF 88 data for a representative busy day.

*Example 8-2 Sample job to report on a single log stream*

```
//IXGRPT1L JOB 'Sample 88 pgm','Stephen Anania',
// MSGLEVEL=(1,1),CLASS=A,NOTIFY=&SYSUID.,MSGCLASS=H
//* *****
//* This jobs formats the SMF88s for a single logstream. *
//* *
//* Unfortunately, the SORT SYSIN does not support symbols, so you *
//* must use ISPF EDIT CHANGE command to set the desired logstream. *
//* Update 'logstream_name' to the name of the logstream you want. *
//* *
//* *****
// SET SMFIN1=#@OPR17.SMFDATA.G0003V00
// SET LIBPRFX=CEE
// SET RPTOBJ=SYS1.SAMPLIB(IXGRPT1L)
//EXTRACT EXEC PGM=IFASMFDP
//SYSUDUMP DD DUMMY
//SYSPRINT DD SYSOUT=*
//SMFG00 DD DISP=SHR,DSN=&SMFIN1
//SMFALL DD DSN=&&SMF88S,UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(CYL,(5,5))
//SYSIN DD *
INDD(SMFG00,OPTIONS(DUMP))
OUTDD(SMFALL,TYPE(88))
/*
/* SORT EACH SMF INPUT DSN BY TIMESTAMP AND LOGSTREAM NAME
//SORT1 EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=&SMF88S,DISP=(OLD,KEEP)
//SORTOUT DD DSN=&&TEMP1,DISP=(NEW,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1))
//SYSIN DD *
OPTION VLSHRT
SORT FIELDS=(133,8,BI,A,
105,26,CH,A)
INCLUDE COND=(23,2,BI,EQ,X'0001',AND,
105,26,CH,EQ,C'logstream_name')
/*
//ASMC EXEC PGM=ASMA90,REGION=OM,PARM='OBJECT,NODECK' *
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR * LIBRARY DSN
//SYSLIN DD DSN=&&OBJ(IXGR1A),SPACE=(TRK,(10,30,10),,ROUND),
// UNIT=3390,DISP=(MOD,PASS),
// DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB) * ASM OBJ
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSIN DD DSN=SYS1.SAMPLIB(IXGR1A),DISP=SHR * ASM SOURCE
/*
/* *****
/* LINK/EDIT, AND EXECUTE OBJECT SAMPLIB MEMBER USING LE LIBRARIES.
/* *****
//LKED EXEC PGM=IEWL,REGION=1024K
//SYSLIB DD DSN=&LIBPRFX..SCEELKED,DISP=SHR * CATALOGUED PROC
//SYSPRINT DD SYSOUT=*
//SYSLIN DD DSN=&RPTOBJ.,DISP=SHR *
// DD DDNAME=SYSIN
//SYSLMOD DD DSN=&&LOADMOD(GO),DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(1024,(50,20,1)) * LOAD MODULE
//SYSUT1 DD DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),
```

```

//          DCB=BLKSIZE=1024
//SYSIN    DD  DSN=*.ASMC.SYSLIN,DISP=(MOD,PASS) * ASM OBJ (IN)
//GO       EXEC PGM=*.LKED.SYSLMOD,COND=(4,LT,LKED),
//          REGION=2048K
//STEPLIB  DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR      * CATALOGUED PROC
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SMFDATA  DD  DSN=*.SORT1.SORTOUT,DISP=(MOD,PASS)
//RPORT88  DD  SYSOUT=*                          * IXGRPT1 OUTPUT

```

---

In the resulting report, select the interval that has the highest *BYT Written to interim storage* value—this identifies the busiest interval. Unlike CF log streams, the *element size* (actually the staging data set CI Size) is sufficiently large that we *do* need to take it into account. So, now we check the “Average Buffer Size” value for that interval. Divide the average buffer size value by 4000 and round up to the next whole number—this is the number of CIs required in the staging data set for each IXGWRITE. Next divide the “# Writes invoked” value by the number of minutes in the SMF interval—this gives us the number of IXGWrites per minute. Now multiply that number by the number of CIs required for each IXGWRITE and that gives us the number of CIs being written to this log stream per minute. Increase this value by 50% to allow for unexpected spikes in activity and control data. The resulting value indicates how much space there should be in the interim storage for this log stream below the HIGHOFFLOAD threshold, assuming we have one offload per minute. The LOWOFFLOAD threshold for funnel-type log streams should always be 0%, so we can ignore that value in these calculations

For example, assume the average buffer size for the busiest interval is 4100, the number of writes invoked is 45000, and the SMF interval is 15 minutes. Dividing the 45000 by 15 minutes gives a write rate of 3000 writes a minute. Next we divide the average buffer size (4100) by 4000 and round up—this gives us a value of two CIs per IXGWRITE. Increasing by 50% brings this up to 3 CIs per IXGWRITE. Multiply this by the number of IXGWrites per minute and we get a value of 9000 CIs per minute. So, the staging data set space below the HIGHOFFLOAD threshold should be 9000 CIs. If the HIGHOFFLOAD threshold for this log stream is 70%, the data set should be sized to be roughly 13000 CIs (9000 / 70%). This would result in an offload every minute during peak times. We will come back to the values to use for HIGHOFFLOAD and LOWOFFLOAD in 8.4.1, “High and low offload thresholds for funnel-type log streams” on page 284.

### **Sizing interim storage for active DASD-only log streams**

For active exploiters, you must use a different methodology to calculate the amount of space in the staging data set part of the log stream. This is because you ideally want all log data to be deleted before it is offloaded to the offload data sets. To achieve this, you need to work with both the amount of space available in the interim storage of this log stream, and also the HIGHOFFLOAD and LOWOFFLOAD thresholds.

Once again, we must use information from the IXGRPT1 report. There are two indicators that you must watch for:

- ▶ The frequency of offload processing. At peak times, there should not be more than one offload every 30 seconds, and *at a minimum* there should be *at least* one offload per SMF interval.
- ▶ The amount of data that is moved to an offload data set. This information is kept in field SMF88LDB in the SMF Type 88 record, and is shown in the field entitled “BYT Written to DASD” in the IXGRPT1 report. For an active log stream, this should always be zero.

To size interim storage for an active log stream, you need to arrive at a size, and LOWOFFLOAD and HIGHOFFLOAD values, that will result in about one offload per minute at peak times and 0 bytes being moved to offload data sets.

To do this, we need some more information from the IXGRPT1 report. We need to know the rate at which IXGWrites are being issued, and the number of CIs used by each IXGWRITE. Using the IXGRPT1 report, select the interval that has the highest “BYT Written to interim storage” value—this identifies the busiest interval. Next we check the *Average Buffer Size* value for that interval. Divide the average buffer size value by 4000 and round up to the next whole number—this is the number of CIs required in the staging data set for each IXGWRITE. Next divide the “# Writes invoked” value by the number of minutes in the SMF interval—this gives us the number of IXGWrites per minute. Now multiply that number by the number of CIs required for each IXGWRITE and that gives us the number of CIs being written to this log stream per minute. Increase this value by 50% to allow for unexpected spikes in activity and control data. The resulting value indicates how much space there should be in the interim storage for this log stream *between the HIGHOFFLOAD and LOWOFFLOAD thresholds*, assuming we have one offload per minute.

We also need to know how often data is being logically deleted from the log stream (using IXGDELETE). If there are no IXGDELETs issued between two consecutive offloads, System Logger has no choice but to move the oldest data from interim storage to an offload data set. For some exploiters, like CICS, you can affect how often an IXGDELETE is done by adjusting the Activity Keypoint Frequency. However, for exploiters where you cannot control this, you need to tune the frequency of offloads by adjusting the interim storage size—the larger the interim storage, the less frequently will an offload be initiated. You want to make sure there is at least one IXGDELETE per offload. You can find the number of IXGDELETs in the interval by summing the “# Deletes w/o DASD write” and “# Deletes w/ write fields”. Divide this value by the number of minutes in the SMF interval to determine the number of IXGDELETs per minute.

So, now we know how many CIs are being written per minute, and how many IXGDELETs are being issued per minute. If you have more than one IXGDELETE per minute, you should aim to have one offload per minute. If you only have one IXGDELETE every x minutes, you should aim to have one offload every x minutes.

To calculate the interim storage size, multiply the number of CIs being written per minute by the number of minutes between offloads. The result is the amount of space in the interim storage between the HIGHOFFLOAD and LOWOFFLOAD thresholds. To get the total interim storage space for this log stream, divide this amount by the difference between the HIGHOFFLOAD and LOWOFFLOAD percents. So, if the amount of data is 3MB, and the HIGHOFFLOAD is set to 80 and LOWOFFLOAD is set to 60, you should have  $3\text{MB} / (80 - 60\%)$ , or 15MB of space in the staging data set for this log stream. We will come back to the values to use for HIGHOFFLOAD and LOWOFFLOAD in 8.4.2, “High and low offload thresholds for active log streams” on page 284.

### 8.3 System Logger duplexing of interim storage

As discussed in “Duplexing log data for CF-Structure based log streams” on page 42, System Logger always keeps two copies of the data that is currently resident in interim storage. The interim storage could be a CF structure or a staging data set (for DASD-only log streams), and the second copy could be a data space or a staging data set (for CF-Structure log streams). We will address CF-Structure log streams and DASD-only log streams separately.

### 8.3.1 CF-Structure log stream

The maximum IXGWRITE rate that a log stream can handle is determined by the placement of the log data interim storage and its duplex copy. A log stream that is duplexing to staging data sets will be limited to roughly the same IXGWRITE rate that can be achieved with DASD-only log streams. All other things being equal, the maximum IXGWRITE rate for a CF log stream that is not using staging data sets is at least one order of magnitude higher than for a DASD-only one.

Bearing these attributes in mind, the best performance can be obtained by ensuring that CF-Structure log streams are duplexed to data spaces, and keeping the associated structure size as small as possible, while still being able to deliver the required levels of performance.

For a CF-Structure log stream, the location of the second copy of the data depends on how you specified STG\_DUPLEX and DUPLEXMODE in the log stream definition. If you specify STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND), the second copy of the data will always be in a staging data set. If you specify STG\_DUPLEX(YES) and DUPLEXMODE(COND), the location of the second copy depends on the volatility of the CF and on whether the CF is in the same failure domain as this z/OS system.

If the CF is volatile, the second copy of the data currently in interim storage will be placed in a staging data set. If the CPC containing the CF has a battery backup, the CF will know that it is non-volatile. If the CPC does not have battery backup, but you have a UPS and are 100% confident that the UPS will not fail, you can issue a command on the CF console to tell it that it is non-volatile. Placing a System Logger structure in a volatile CF will cause all log streams in that structure that have specified DUPLEXMODE(COND) to be duplexed to staging data sets on DASD, impacting performance at higher logging rates.

If the z/OS is in the same failure domain as the CF (that is, they reside in the same CPC), the second copy of the data will again be placed in the staging data set. If two z/OSs in two CPCs are connected to the same log stream, it is possible that one of the z/OSs will be in the same failure domain as the CF, but the other will not. In this case, one of the System Loggers will be writing the second copy of the data to a staging data set and the other will be using a data space.

If neither of these conditions apply, the second copy of the data will be placed in a data space.

And as discussed in 2.8.3, “Other recovery processes and errors” on page 72, the data space is used to repopulate the structure as part of a structure rebuild operation. Therefore, having a large amount of log stream data in a data space can result in significant paging during certain recovery processing. This is another reason for not over-sizing the interim storage for a log stream—the larger the interim storage defined for a log stream, the larger the data space that is required to hold the duplex copy of that data.

**Tip:** If at all possible, configure your System Logger environment so that the second copy of the data is kept in a data space. This provides the best configuration relative to performance throughput, although not necessarily from a recovery situation, since using staging data sets can offer better benefits. For any log stream you can find the current location of the second copy by issuing the following command:

```
D LOGGER,C,LSN=log_stream_name,DETAIL
```

If the DUPLEXING: line in the response contains LOCAL BUFFERS, the data is being kept in the data space. Remember that you must issue this command on each system that is connected to the log stream in question.

If you are using SMF Type 88 records to check this, the second bit of the SMF88LFL field indicates whether that log stream used a staging data set during that SMF interval.

### 8.3.2 DASD-only log streams

With DASD-only log streams, the second copy of the log data is always kept in a data space—this is something that you have no control over.

However, you can, indirectly, control the *size* of the data space. Remember that all the log data in a staging data set at any one time will also exist in a data space. So, the larger the staging data set, the larger the data space needs to be. As for CF log streams, we do not recommend specifying the interim storage for a DASD-only log stream any larger than necessary.

### 8.3.3 Sizing duplex copy of CF log streams

When sizing the interim storage for your log streams, it is important to bear in mind the requirements for the second copy of the log data. There are a number of factors that make this a little complex.

If the log stream in question is a CF one, the first thing to consider is how many log streams will normally reside in the structure. The structure storage is divided equally between all the connected log streams in the structure. So, if you have a 46 MB structure with 10 log streams, each log stream will have roughly 4 MB of storage in the CF.

Because the unit of storage in the CF is more granular (256 or 512 bytes) than in the staging data sets (4 KB), it is likely that a staging data set will need to be larger to hold the same amount of log data as a structure. For example, if you were writing small log records (100 bytes), the staging data set would need to be 16 times larger than that log stream's portion of the CF structure to hold the same number of records (because each record would take up 256 bytes in the CF, but 4096 bytes in the staging data set).

To get optimum performance from a CF log stream, it is important that offloads are initiated because the CF interim storage hit the HIGHOFFLOAD threshold, and not because the staging data set hit the threshold first. When sizing the staging data sets, you should use the formula we described in “Sizing interim storage for funnel-type DASD-only log streams” on page 278. You should also monitor for non-zero values in the STG THLD field in the IXGRPT1 report. If you encounter instances of non-zero values, increase the STG\_SIZE specification on the log stream definition in the LOGR policy by a small amount to make the staging data set larger, and continue to monitor.

## 8.4 Setting HIGHOFFLOAD and LOWOFFLOAD thresholds

The setting of the high and low offload thresholds have a significant impact on the performance and availability of the log stream. The values that you specify differ, depending on whether the log stream connector is a funnel-type or an active one.

### 8.4.1 High and low offload thresholds for funnel-type log streams

The low offload threshold for a funnel-type log stream should *usually* be zero. Most situation will be satisfied by having the data offloaded to a low offload value of zero.

The setting of the high offload threshold usually can be calculated and adjusted according to your environment. As a general rule, a high offload threshold of 70% should provide acceptable performance and availability. The most important thing is that it should be possible to move all the data below HIGHOFFLOAD to the offload data sets in less time than it would take the application to fill the space between HIGHOFFLOAD and the interim storage filling completely.

This is shown in Figure 8-1. In this example, the structure is 50 MB, and the HIGHOFFLOAD threshold is set to 70%, meaning that there is 15 MB above the threshold. The application writing to this structure is generating 450 KB of log data per second, meaning that it would take 33 seconds to fill up the structure. In order to provide acceptable performance, it must be possible to offload the 35 MB of data below the HIGHOFFLOAD threshold in less than 33 seconds. If it turns out that you cannot offload the data in that time, you can decide to decrease the HIGHOFFLOAD threshold, meaning that there is less data to move every time HIGHOFFLOAD threshold is reached, and also that there is more storage above the threshold, meaning that it will take longer to fill the structure.

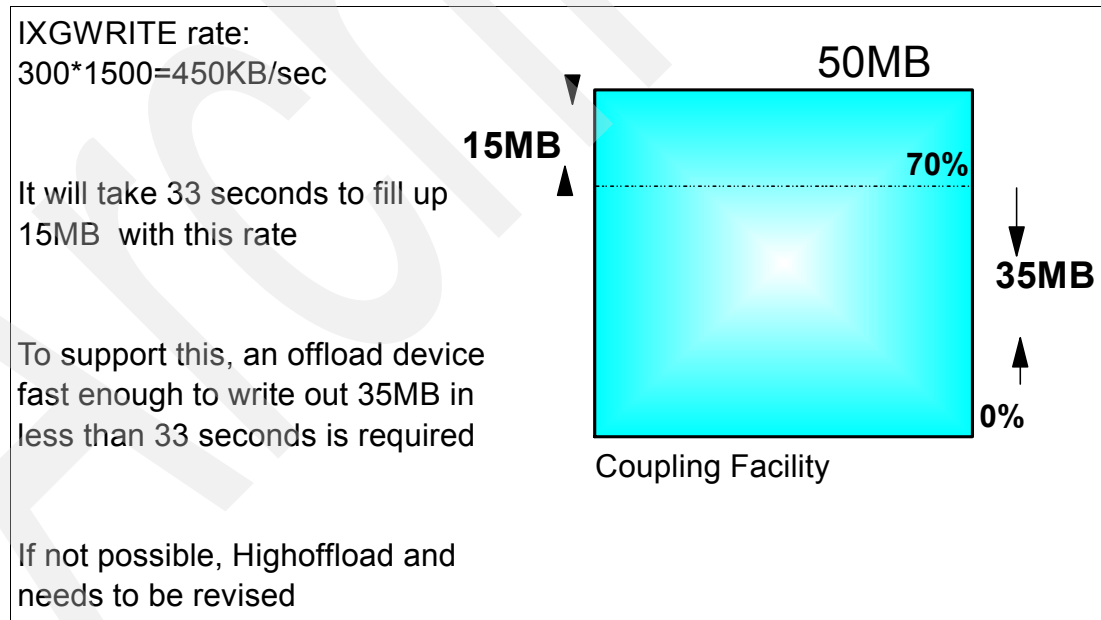


Figure 8-1 Calculating High Threshold value

### 8.4.2 High and low offload thresholds for active log streams

Unlike funnel-type log streams, you specifically do *not* want to move all data out of interim storage. Rather, you want to keep data in interim storage for as long as possible, within the constraint of always having to have available space in interim storage for new writes.

The HIGHOFFLOAD threshold for an active log stream should not be higher than 80%. However, there is no good reason for specifying it any lower than this value.

There are two schools of thought about the value that should be used for the LOWOFFLOAD threshold. One holds that it should be set to 60%, striking a balance between keeping as much data in interim storage as possible, while at the same time ensuring that each offload frees up enough data to ensure that another offload will not be required too soon. The other holds that LOWOFFLOAD should be set as high as 75%. The reason for this is that it minimizes the amount of processing required by the offload and increases the amount of data that can be kept in interim storage. However, measurements have shown that any difference in performance between the two values falls within the normal margin of error.

## 8.5 Other performance recommendations

The remaining performance-related recommendations apply equally to active and funnel-type log streams and are documented in this section.

### 8.5.1 Which CF log streams to place in the same structure

There are two attributes of a log stream that should be borne in mind when deciding which log streams should reside in a given structure. In addition, it is recommended that you not place more than 10 log streams in a given structure. You can enforce this by specifying LOGSNUM(10) when you define the structure in the LOGR policy.

The first attribute is the ratio between the MAXBUFSIZE specified for the structure and the actual average buffer size of the log stream. The actual buffer size for each log stream can be found in the field entitled “AVERAGE BUFFER SIZE” in the IXGRPT1 report. This ratio determines how many elements are needed for each log block, and therefore the entry-to-element ratio. This was discussed in “The entry-to-element ratio” on page 26. If the log streams in a structure have very different average buffer sizes, it is likely that the entry-to-element of the structure will only suit a subset, or maybe even none, of the log streams in the structure.

The average buffer size of a log stream can vary over time, so this is not an exact science, however, try to avoid placing log streams with consistently very large average buffer sizes in the same structure as ones with consistently very small ones.

The other attribute to be considered is the volume of data being written to the log stream. The number of bytes written to the log stream over an SMF interval is reported in field “BYT WRITTN BY USERS IXGWITES” in the IXGRPT1 report. Remember that the elements in a structure are divided equally between all connected log streams in that structure. So, if you have a very busy and a very idle log stream in the same structure, the busy one is likely to be short of storage, doing very frequent offloads, while the other log stream will have far more storage than it requires and therefore hardly ever doing an offload.

Traditional wisdom for placing DASD data sets is that busy data sets should be separated from each other and placed on volumes with idle data sets. However, for optimum CF log stream performance, log streams with high write rates should be placed in the same structure—one that has a reasonably large size. And log streams with low writes should be placed together in another structure, presumably one with a lot less storage.

## 8.5.2 System Logger DASD performance

System Logger uses three types of data sets: its own CDSs, staging data sets, and offload data sets. While the CDSs are not normally very busy, during recovery processing they can get quite busy. Therefore, to ensure timely recovery from failures, the LOGR CDSs should be placed on high performance DASD, along with the other sysplex-related CDSs.

The staging data sets associated with busy log streams can potentially be a bottleneck to System Logger performance. Remember that IXGWRITES are synchronous—the requestor will not get a response from System Logger until the I/O to the staging data set either completes or fails. Therefore, at least the staging data sets associated with the busier log streams should be placed on high performance devices. You can use your SMS ACS routines to ensure this happens.

The last type of data sets are the offload ones. Generally speaking, the performance of these data sets is not as critical as the staging data sets. What is more important is that they are placed on volumes that are not subject to RESERVEs as this can delay allocation processing when System Logger is trying to move data to an offload data set from interim storage. Therefore, when placing these data sets (in your ACS routines), attempt to select devices with reasonable performance that are unlikely to be the targets of a RESERVE.

In addition to placement of the staging data sets, the CI Size of the data sets can have an affect on System Logger performance. While the staging data sets must have a CI Size of 4 KB, the offload data sets should have a CI Size of 24 K. The best way to effect this is to use a data class for all offload data sets that is defined with this CI Size.

## 8.5.3 Indicators of potential problems

There are a number of fields in the IXGRPT1 report that you should monitor that warn you of potential sub-optimal performance.

The first is significant number of “Type 3” writes for any CF log stream. These are a result of an IXGWRITE to a log stream where there are already more than 90% of the elements are in use. This is an indicator of either a problem in offload processing, or else that the interim storage is significantly undersized for this log stream.

Another field to monitor is DASD SHIFTS, particularly in relation to the number of offloads. Assuming that offloads are being initiated reasonably frequently, the number of DASD SHIFTS (that is, the number of times a new offload data set had to be allocated) should be a small percentage of the number of offloads. If you have a significant number of DASD SHIFTS, it is an indicator that the offload data sets are much smaller than they should be (possibly because an appropriate LS\_SIZE was not specified on the log stream definition in the LOGR policy). In addition, for an active log stream, where you want to avoid the use of offload data sets completely, the presence of DASD SHIFTS is an indicator that either the log stream interim storage is too small, or else there is a problem with the application resulting in log records being held open for much longer than they should be. This is discussed in “IXGRPT1 observations and possible actions” on page 195.

## 8.6 Using System Logger performance data

The following section gives a description of the tools available to collect data regarding System Logger and how to interpret the values.

One of the challenges when tuning System Logger is that there is no online monitor that provides information at the log stream level. While RMF provides some information, it is only



at the address space or structure level, not at the log stream level. To get log stream level information, you must post-process the SMF Type 88 records. Methods for doing this will be discussed below.

### 8.6.1 RMF: Analysis of the CF structure service times


For CF-Structure log streams, the RMF Coupling Facility Activity Report can provide important information about structure usage and service times. However, this report only provides information at the structure level—you must determine the number of log streams associated with the structure in question from either the SMF Type 88 data, or from the log stream definitions.

#### ***RMF Coupling Facility Activity Report***

```

-----
COUPLING FACILITY NAME = CF103
TOTAL SAMPLES (AVG) = 299 (MAX) = 299 (MIN) = 299
-----
                                COUPLING FACILITY USAGE SUMMARY
-----
STRUCTURE SUMMARY
-----

```

TYPE	STRUCTURE NAME	STATUS CHG	ALLOC SIZE	% OF		AVG REQ/SEC	LST/DIR TOT/CUR	DATA ELEMENTS TOT/CUR	LOCK ENTRIES TOT/CUR	DIR REC/DIR REC XI'S
				CF STORAGE	# REQ					
LIST	DSNZPLEX_SCA	ACTIVE	32M	1.6	0	0.00	43K	86K	N/A	N/A
	LOG_JG2_5M	ACTIVE	8M	0.4	1585K	5284.3	128	819	N/A	N/A
							1554	3143	N/A	N/A

```

-----
PROCESSOR SUMMARY
-----
COUPLING FACILITY      2064      MODEL 100      CFLEVEL 12
AVERAGE CF UTILIZATION (% BUSY)      20.2      LOGICAL PROCESSORS:  DEFINED  1      EFFECTIVE  1.0
-----
                                COUPLING FACILITY ACTIVITY
-----
z/OS V1R4              SYSPLEX WSCZPLEX              DATE 05/28/2003              INTERVAL 005.00.000
                          RPT VERSION V1R2 RMF              TIME 16.30.00              CYCLE 01.000 SECONDS
-----
                                COUPLING FACILITY STRUCTURE ACTIVITY
-----
STRUCTURE NAME = LOG_JG2_5M      TYPE = LIST      STATUS = ACTIVE
# REQ      -----      REQUESTS      -----      DELAYED REQUESTS -----
SYSTEM TOTAL      #      % OF -SERV TIME (MIC)-      REASON      #      % OF ---- AVG TIME (MIC) ----
NAME      AVG/SEC      REQ      ALL      AVG      STD_DEV      REQ      REQ      /DEL      STD_DEV      /ALL
SYSD      1585K      SYNC      1458K      91.9      43.7      54.8      NO SCH      0      0.0      0.0      0.0      0.0
          5284      ASYNC      128K      8.1      323.5      406.6      PR WT      0      0.0      0.0      0.0      0.0
          CHNGD      0      0.0      INCLUDED IN ASYNC      PR CMP      0      0.0      0.0      0.0      0.0
          DUMP      0      0.0      0.0      0.0      0.0
-----

```

Figure 8-2 RMF Coupling Facility Activity report

In the sample report shown in Figure 8-2, the structure size for LOG\_JG2\_5M has been altered to 8M, which is 0.4% of the total CF storage. The structure was defined with an INITSIZE of 5M, but altered to 8M due to an excessive number of structure full conditions while running the workload. This structure did 100% of the requests to the CF in the 5 minute interval shown. The average request rate was 5284 per second.

System Logger log streams are reported in the LIST section of the Usage Summary part of the report. Under the LST/DIR column there are two lines per structure. The first line gives the

total number of *entries* in the structure, and the second line shows the number currently in use. In the Data Elements column, the first line gives the total number of data *elements* in the structure, the second line gives the number currently in use. The entries are in a common pool for the structure (the entries for all log streams come from the same pool). Dividing the number of data elements (11K) by the number of entries (5725) gives an entry-to-element ratio of 1:2. The entry-to-element ratio is dictated by the log stream with the largest percentage of records in the structure. If the number of entries in use reaches 90% of the total number of entries available for the structure, System Logger will force an offload of *all* log streams in the structure.

Notice the current usage (second row of data) indicates a 1:3 ratio (3143 in-use elements/1554 in-use entries), rounded up to the next whole number. As noted above, the number of entries (5725) and elements (11 K) in the structure are in a 1:2 ratio. With a MAXBUFSIZE set to 64000, the element size is 256 and consequently, because of the 1:2 ratio, this indicates an expected buffersize of 512 bytes. However, the current usage of 1:3 indicates at least one log stream is writing records of at least 750 bytes.

An important point to remember is the number of data elements is divided equally among the connected log streams. So, if the number of data elements in the structure is 11K, and there are three log streams connected to the structure, each log stream is allocated 3.66K data elements.

The Coupling Facility Activity report also provides information about the request activity for the structure. From the SYSD system, there were 1585K requests, giving an average of 5284 per second. 1458K of the requests were synchronous with average service times of 43.7 microseconds.

The average, or *mean*, represents the middle in the distribution of a set of individual measurements. The standard deviation measures the spread or variation of the individual measurements on either side of the average, 66% of all observations lie within plus or minus one standard deviation. 95% of all observations lie within plus or minus 2 standard deviations.

The report shows an average SYNC time of 43.7 microseconds with a standard deviation of 54.8 microseconds for the population of SYNC requests. In this case, two standard deviations to the negative side of the average would be less than zero. Consider 0 to +2 standard deviations on the positive side of the average defines the range where 95% of the requests will occur. Therefore, 95% of all SYNC requests for this test would lie between 0 and 153.3 ( $2 * 54.8 + 43.7$ ) microseconds. If the standard deviation is large (for example in the thousands of microseconds), it indicates some portion of the CF configuration is non-responsive, causing large variability in individual measurements.

The most frequently seen reason for a non-responsive CF is the use of shared CF CPs, especially if Dynamic CF Dispatching (DYNDISP=YES) is enabled for the CF LPAR. Looking at the RMF CF Usage Summary report, whenever the number of logical processors defined is greater than the number of effective logical processors, the configuration may be seeing performance issues. For production CICS regions, if you must use a CF with shared CPs, turning Dynamic CF Dispatching off (DYNDISP=NO) is strongly recommended. The DYNDISP command is entered on the CF console.

Another point of caution—if the CF is actually an LPAR in the same CPC as the z/OS image, and the LPARs share CPs, *all* SYNC requests will be converted to ASYNC requests. Neither CICS nor the System Logger have control (or knowledge) of the change, and the reports will show the requests as SYNC, but the service times will be elongated.

It's also good to check for delayed requests. For example, if the number of NO SCH (that is, no subchannel), is greater than 10% of the total number of requests for the structure, you should consider adding more CF Link capacity, or reducing the load on the existing links.

Figure 8-3 contains sample JCL to produce the Coupling Facility Activity report.

```
//CFACT JOB (????,????), 'JIM GRAUEL',MSGLEVEL=(1,1),
// CLASS=A,MSGCLASS=0,NOTIFY=GRAUEL,REGION=0M
/*JOBPARM SYSAFF=SYSD
//*****
/** FORMAT THE CF ACTIVITY REPORT ****
//*****
/* PROVIDE THE SMF DATASET NAME *
// SET SMFIN1=GRAUEL.PERFTST.R222.RED1R.P1.MAY31.ELEVEN03
//*****
//S1 EXEC PGM=ERBRMFPP,REGION=1000K
//MFMSGDS DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//MFPINPUT DD DSN=&SMFIN1,DISP=(OLD,KEEP)
//SYSIN DD *
SYSOUT(0)
SYSRPTS(CF) /*COUPLING FACILITY */
//NULL DD *
```

Figure 8-3 Sample JCL to produce an RMF CF Activity Report

## 8.6.2 RMF: Workload Activity Reports

(Use: Analysis of RMF data)

RMF provides a wealth of information which is invaluable in the resolution of performance problems. This information can be used to measure the impact of log stream usage and definition changes.

When using z/OS WLM in goal mode, it's possible to define a report class for each CICS region and the System Logger address space. The report classes for regions are in addition to the report classes for transactions as discussed in "DFHLOG and DFHSHUNT example" on page 177.

Each CICS region and System Logger address space should be defined in a separate report class in order to understand the results of each change made during the log stream tuning process.

The same information can be gathered using a Report Performance Group if WLM is being used in compatibility mode. For guidance in defining a compatibility mode Report Performance Group, refer to the section entitled "Defining Service Classes and Performance Goals" in *z/SO MVS Planning: Workload Management, SA22-7602*.

Figure 8-5 on page 291 contains a WLM Workload Activity Report, which presents data collected for report classes RIYOT1 (a CICS region) and RLOGGER (the System Logger address space). Report classes are defined using the WLM ISPF panels.

Sample JCL to produce the Workload Activity report is shown in Figure 8-4.

```
//RCLASS JOB (????,????), 'JIM GRAUEL',MSGLEVEL=(1,1),
// CLASS=A,MSGCLASS=0,NOTIFY=GRAUEL,REGION=OM
/*JOBPARM SYSAFF=SYSD
//*****
/**      FORMAT THE REPORT CLASSES          ****
//*****
/* PROVIDE THE SMF DATASET NAME          *
// SET   SMFIN1=GRAUEL.PERFTST.R222.RED1R.P1.MAY31.ELEVEN03
//*****
//RMFPP EXEC PGM=ERBRMFPP,REGION=OM
//MFPMMSGDS DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//MFPINPUT DD DSN=&SMFIN1,DISP=(OLD,KEEP)
//SYSIN DD *
SYSOUT(0)
SYSRPTS(WLMGL(RCLASS(RLOGER))) /* REPORT CLASS FOR THE LOGGER*/
SYSRPTS(WLMGL(RCLASS(RIYOT1))) /* REPORT CLASS FOR IYOT1 */
DINTV(0005)
SUMMARY(INT)
/*
```

Figure 8-4 JCL to produce RMF Workload Activity report

The report interval is listed in the START and END times at the top of the report page. A word of caution—the minimum interval is defined by the INTVAL() parm in the SMFPRMxx member of SYS1.PARMLIB. In the samples collected, the interval was set to 5 minutes as follows:

```
INTVAL(05)          /* SMF GLOBAL RECORDING INTERVAL */
```

It's important to ensure the SMF 70 to 79 records are being collected, along with the CICS 110 records. The records that are to be collected are also defined in the SMFPRMxx member.

```
SUBSYS(STC,EXITS(IEFACTRT),INTERVAL(SMF,SYNC),
        TYPE(0,30,70:79,88,89,90,99,110,245))
SUBSYS(OMVS,NOEXITS,INTERVAL(SMF,SYNC),
        TYPE(0,30,70:79,88,89,90,99,110,245))
```

When the reports are formatted, it is possible to report a larger interval than was specified in the SMFPRMxx member, by using the DINTV parm for the ERBRMPFF utility. However, the smallest interval which can be reported is the value specified for INTVAL.

The following fields should be noted in the reports:

- ▶ Under TRANSACTIONS:
  - MPL - Multiprogramming level—the number of address spaces active in this service/report class during the interval. The value should be one for the most accurate measurements for a given region.
- ▶ Under DASD I/O:
  - SSCHRT - Number of start subchannels (SSCH) per second in the reported interval
  - RESP - Average DASD response time (in milliseconds)
- ▶ Under SERVICE RATES:
  - TCB - CPU seconds accumulated in TCB mode during the collection interval
  - SRB - CPU seconds accumulated in SRB mode during the collection interval
  - APPL% - Percentage of a engine (CP) used during the collection interval

- ▶ Under PAGE-IN RATES:

SINGLE - The average rate at which pages are read into main storage

- ▶ Under STORAGE:

AVG - Average number of central and expanded storage frames allocated to ASIDs in the report class

Refer to *z/OS V1R4.0 RMF Report Analysis*, SC33-7991, for more information about this report.

WORKLOAD ACTIVITY													
z/OS V1R2		SYSPLEX WSCZPLEX		START 01/01/2002-23.15.00		INTERVAL 000.04.59		MODE = GOAL		PAGE 1			
		RPT VERSION V1R2 RMF		END 01/01/2002-23.19.59									
POLICY ACTIVATION DATE/TIME 11/21/2001 12.21.14													
-----													
CLASS (ES) REPORT													
REPORT BY: POLICY=WLMPOP													
REPORT CLASS=RIYOT1													
DESCRIPTION =CICS Report Class for Jim G.													
TRANSACTIONS	TRANS.	TIME	HHH.MM.SS.TTT	--DASD	I/O--	--SERVICE--	SERVICE RATES-	PAGE-IN RATES					
-----STORAGE-----													
AVG	1.00	ACTUAL	0	SSCHRT	11.4	IOC	343	ABSRPTN	631	SINGLE	0.0	AVG	8517.43
MPL	1.00	EXECUTION	0	RESP	1.9	CPU	9348	TRX SERV	631	BLOCK	0.0	TOTAL	8517.43
ENDED	0	QUEUED	0	CONN	1.6	MSO	175902	TCB	0.8	SHARED	0.0	CENTRAL	8517.43
END/S	0.00	R/S AFFINITY	0	DISC	0.1	SRB	3566	SRB	0.3	HSP	0.0	EXPAND	0.00
#SWAPS	0	INELIGIBLE	0	Q+PEND	0.2	TOT	189159	RCT	0.0	HSP MISS	0.0		
EXCTD	0	CONVERSION	0	IOSQ	0.0	/SEC	631	IIT	0.0	EXP SNGL	0.0	SHARED	1.00
AVG ENC	0.00	STD DEV	0					HST	0.0	EXP BLK	0.0		
REM ENC	0.00							APPL %	0.4	EXP SHR	0.0		
MS ENC	0.00												
REPORT BY: POLICY=WLMPOP													
REPORT CLASS=RLOGER													
DESCRIPTION =Report for System Logger													
TRANSACTIONS	TRANS.	TIME	HHH.MM.SS.TTT	--DASD	I/O--	--SERVICE--	SERVICE RATES-	PAGE-IN RATES					
-----STORAGE-----													
AVG	1.00	ACTUAL	0	SSCHRT	12.3	IOC	1	ABSRPTN	8	SINGLE	0.0	AVG	5115.59
MPL	1.00	EXECUTION	0	RESP	1.1	CPU	85	TRX SERV	8	BLOCK	0.0	TOTAL	5115.59
ENDED	0	QUEUED	0	CONN	0.8	MSO	1181	TCB	0.0	SHARED	0.0	CENTRAL	5115.59
END/S	0.00	R/S AFFINITY	0	DISC	0.1	SRB	1085	SRB	0.1	HSP	0.0	EXPAND	0.00
#SWAPS	0	INELIGIBLE	0	Q+PEND	0.2	TOT	2352	RCT	0.0	HSP MISS	0.0		
EXCTD	0	CONVERSION	0	IOSQ	0.0	/SEC	8	IIT	0.0	EXP SNGL	0.0	SHARED	0.00
AVG ENC	0.00	STD DEV	0					HST	0.0	EXP BLK	0.0		

Figure 8-5 Sample Workload Activity report

### 8.6.3 SMF Type 88 records and IXGRPT1 program

The SMF 88 data fields provide the data to evaluate the operation of log streams managed by the System Logger.

The only generalized interface to System Logger performance information is the SMF Type 88 records produced by System Logger. One SMF Type 88 record is created each interval for each log stream defined in the LOGR policy. IBM provides a sample program in SYS1.SAMPLIB called IXGRPT1 to format the important fields in these records for you. The first step in performing the analysis is to collect the Type 88 records into a data set using JCL similar to that shown in Example 8-3.

Example 8-3 Sample JCL to extract SMF Type 88 records

```
//SMFDUMP JOB (0,0),'SMF DUMP JOB ',MSGCLASS=X,MSGLEVEL=(1,1),
// CLASS=A,NOTIFY=&SYSUID
```

```

/*JOBPARM SYSAFF=#0$2
/* -----
/* THIS JOB WILL COPY THE SMF TYPE 88 RECORDS FROM THE
/* SMF DATA SET TO THE DATA SET SPECIFIED ON THE
/* DUMPOUT DD STATEMENT
/* -----

//STEP1 EXEC PGM=IFASMFDP
//DUMPIN DD DISP=SHR,DSN=SYS1.#0$2.MAN1 <-----
//DUMPOUT DD DSN=KYNEF.SMF88.OUT,
//          DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(500,50),RLSE),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
          OUTDD(DUMPOUT,TYPE(88))
          INDD(DUMPIN,OPTIONS(DUMP))
//NULL DD *

```

---

It is important to understand that System Logger activities can span adjacent intervals. For example, data may be written (SMF88LWB) in the first interval and deleted (SMF88SIB) or offloaded in the second.

When the IXGRPT1 program is used to format the Type 88 records, the timestamp reflects the *end* of the interval, not the beginning as in other SMF reports.

**Attention:** The timestamps in the reports are given in Greenwich Mean Time (GMT) not local time.

Each SMF 88 record reports System Logger activity for one log stream or structure. SMF record type 88 has the following subtypes:

- ▶ Subtype 1 - Reports log stream usage. The records contain the information required for log stream or structure diagnosis and performance analysis.
- ▶ Subtype 11 - Records CF structure alter activity. Subtype 11 records are produced when the System Logger changes the entry-to-element ratio for CF-Structure log streams.

Note the IXGRPT1 program used to format and report on the SMF Type 88 data does *not* have the ability to summarize at an interval larger than the interval used for data collection (the INTVAL value specified in the current SMFPRMxx).

SMF 88 subtype 11 records are also recorded if the size of the structure is altered using a SETXCF ALTER command, for example:

```
SETXCF START,ALTER,STRNM=LOG_JG2_5M,SIZE=10240
```

An SMF 88 subtype 11 record is written at the completion of the SMF interval, and when the application disconnects from the log stream.

SMF Type 88 records are mapped using the IXGSMF88 macro found in SYS1.MACLIB.

The IXGRPT1 program is supplied as PL/I source in SYS1.SAMPLIB. In addition, member IXGRPT1J is supplied as self documenting JCL to link-edit and execute a pre-compiled version of the utility, IXGRPT1L which is also supplied in SYS1.SAMPLIB.

The utility reads the SMF Type 88 records from a sequential data set and formats the data by log stream within SMF interval.

SYSTEM LOGGER ACTIVITY REPORT (IXGRPTRA)																							
--LOGSTREAM NAME-----	STRUCTURE NAME--	BYT WRITTN		BYT WRITTN		BYT WRITTN		#WRITES	---# WRITES COMPLETED---			AVERAGE											
		BY USERS	TO INTERIM	TO INTERIM	TO DASD	(INVOKED)	INVOKED		TYPE1	TYPE2	TYPE3		BUFFER										
		IXGWITES	STORAGE	STORAGE	(INVOKED)							SIZE											
		BYT DELETD	# DELETES	BYT DELETD	# DELETES	-----EVENT-----																	
		W/O DASD	W/O DASD	W/O DASD	W/DASD	OFF-LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	ST FULL	RE-BLD											
01/30/98 0:46:00 AM	(SMF INTERVAL	'AFEFC71BAFCC7509'X)																					
REESMK.IYK95.DFHJ01	LOG_SYSTEST_006	416	LWB	512	SWB	456	LDB	1	LWI	1	SC1	0	SC2	0	SC3	416C*							
		0	SIB	0	SII	416	SAB	1	SAI	1	EO	1	EDS	0	OESF	0	OEF	0	ETT	0	ETF	0	ERI
01/30/98 0:50:46 AM	(SMF INTERVAL	'AFEFC82CDD3FC400'X)																					
REESMK.IYK95.DFHSHUNT	LOG_GENERAL_008	0		0		0		0		0		0		0		0		0		0		0	
01/30/98 0:50:54 AM	(SMF INTERVAL	'AFEFC833F7006C07'X)																					
REESMK.IYK95.DFHLOG	LOG_GENERAL_005	1112820		1135616		467910		233		230		3		0		4776							
		631323		172		470196		57		2	1	0	0	0	0	0							
01/30/98 0:56:51 AM	(SMF INTERVAL	'AFEFC988410ADE08'X)																					
REESMK.IYK95.DFHJ01	LOG_SYSTEST_006	416		512		456		1		1		0		0		416							
		0		0		416		1		1	0	0	0	0	0	0							
01/30/98 1:13:54 AM	(SMF INTERVAL	'AFEFC58369C4A09'X)																					
REESMK.IYK95.DFHSHUNT	LOG_GENERAL_008	0		0		0		0		0		0		0		0							
		0		0		0		0		1	0	0	0	0	0	0							
01/30/98 1:14:00 AM	(SMF INTERVAL	'AFEFC5DDE02CD07'X)																					
REESMK.IYK95.DFHLOG	LOG_GENERAL_005	3100472		3123200		1182400		207		193		13		1		14978							
		1875985		145		1145356		60		3	3	0	0	0	0	0							
05/11/98 2:21:00 PM	(SMF INTERVAL	'B06F79A6457A6303'X)																					
IYCUZC07.DFHLOG	*DASDONLY*	19444326		28229632		16380778		3894		0		0		0		4993							
		4694016		635		23736320		3318		90	3	0	0	391	1	0							
IYCUZC07.DFHSHUNT	*DASDONLY*	350683		1486848		0		363		0		0		0		966							
		0		0		0		0		0	0	0	0	0	0	0							

Figure 8-6 Sample IXGRPT1 output

Figure 8-6 provides a sample IXGRPT1 output which includes DASD-only and CF-Structure log streams, with the information for the first log stream annotated to show which part of the SMF 88 record each field comes from.

Notice the timestamp—this is related to the SMF interval, but unlike most other reports based on SMF data, the timestamp reflects the *end* of the interval *not* the start.

You notice the report layout takes some getting used to. Each “line” of the report is actually two lines long. Similarly, the column headings are two lines long. For example, taking the right-most column in the report, the heading says “AVERAGE BUFFER SIZE ----- RE- BLD”. This means in the right-most column, the “first” line (the one which contains the log stream name) contains the average buffer size for this log stream, and the “second” line contains the number of times the structure was rebuilt. Although this is a bit confusing at first, you will soon get used to it.

### 8.6.4 IXGRPT1 Field Summary and IXGSMF88 Cross Reference

The following information is intended to provide simple definitions for the report fields and cross reference the report columns to the actual SMF Type 88 data fields. Remember this report only represents information for one system. If you have a log stream which has connectors on multiple systems (forward recovery log, for example), you must view the reports from all systems to get the total view of the activity against the log stream.

The SMF Type 88 fields that are used in the IXGRPT1 report, and the meaning of those fields are as follows:

<b>LWB</b>	Bytes written by IXGWRITE commands.
<b>SWB</b>	Bytes written to interim storage - includes control info.
<b>LDB</b>	Bytes written to offload data sets - includes control info.
<b>LWI</b>	Number of IXGWRITE commands.
<b>SC1/2/3</b>	Number of IXGWrites of each category.
<b>C*</b>	Calculated field, not included in SMF record.
<b>SIB</b>	Bytes deleted by IXGDELET where record was still in interim storage.
<b>SII</b>	Number of blocks deleted by IXGDELET.
<b>SAB</b>	Number of bytes deleted because data was offloaded to DASD.
<b>SAI</b>	Number of blocks deleted after the data had been offloaded to DASD.
<b>EO</b>	Number of offloads.
<b>EDS</b>	Number of DASD shifts.
<b>ESF</b>	Number of times IXGWRITE got "structure full".
<b>EFS</b>	Number of times offload was done because 90% of total structures list entries were in use.
<b>ETT</b>	Number of times staging data set hit threshold.
<b>ETF</b>	Number of times staging data set filled up.
<b>ERI</b>	Number of structure rebuild events initiated.

A more comprehensive description of each field is as follows:

- ▶ **Logstream name (SMF88LSN)**  
Log stream name.
- ▶ **Structure Name (SMF88STN)**  
Name of the structure used for this log stream. For a DASD-only log stream, this field will show \*DASDONLY\*.
- ▶ **BYT Written by users IXGWrites (SMF88LWB)**  
This is the total number of bytes passed to System Logger on IXGWRITE requests over the SMF interval. It does not contain any "hot air"<sup>2</sup> assumptions or rounding to the interim storage medium's storage boundary.  
The computation is the same for Coupling Facility log streams and DASD-only log streams.
- ▶ **BYT Written to interim storage (SMF88SWB)**  
The amount of storage used in the "interim" storage portion of the log stream.  
This is the total number of bytes passed to System Logger on IXGWRITE requests (from SMF88LWB), *plus* the "hot air" used to fill out the storage medium increments. It helps to reveal the amount of "hot air" space in the interim storage medium.  
For CF log streams, this value includes the amount of element space, rounded up to the element boundary, (either 256 or 512 byte) for each IXGWRITE. For CF log streams that

<sup>2</sup> "Hot air" is the difference between the increments of space in the interim storage (for example, 4096 byte CIs in the staging data sets) and the amount of data passed on the IXGWRITE request. For example, if 1096 bytes of data were passed on IXGWRITE for a DASD-only log stream, the "hot air", or empty space, would be 3000 bytes.



are duplexed to staging data sets, this field still represents the amount of space in the CF that is used—space usage for the staging data sets is *not* reported in this case.

For DASD-only log streams, this value includes the amount of staging data set CI space (rounded up to the CI boundary (4096 bytes) for each IXGWRITE) required to hold the log data written to the log stream, plus System Logger's staging data set block management control information.

A point of interest - if the same data were written to a DASD-only log stream and a CF-Structure log stream, SMF88SWB will normally be larger for the DASD-only because of the larger storage increments for DASD-only log streams (4096 bytes versus 256 or 512 bytes).

▶ **BYT Written to DASD (SMF88LDB)**

The number of bytes written to the offload data set(s). This includes System Logger control information (40 bytes per log block, at the time of writing). This data is *not* rounded to a CI boundary—in other words, while each staging data set CI only contains one log block, an offload data set CI can potentially contain a number of log blocks.

▶ **# Writes invoked (SMF88LWI)**

The number of IXGWrites issued by the connectors on this system.

▶ **# Writes Completed**

– **TYPE1 (SMF88SC1)**

The number of IXGWrites which completed normally while the log stream had not yet reached its HIGHOFFLOAD threshold and less than 90% of elements were in use. Prior to APAR OA03055, this field is only used for CF-Structure log streams. With the application of OA03055, this field will also contain data for DASD-only log streams.

– **TYPE2 (SMF88SC2)**

The number of IXGWrites that completed when the log stream was at or above the HIGHOFFLOAD threshold value, but less than 90% of elements for the log stream were in use. Prior to APAR OA03055, this field is only completed for CF-Structure log streams. With the application of OA03055, this field will also contain information for DASD-only log streams.

– **TYPE3 (SMF88SC3) - applies only to CF-Structure log streams.**

The number of writes completed when 90% or more of elements for the log stream are in use. This field is not applicable to DASD-only log streams.

This field should be zero for best performance.

▶ **Average Buffer size**

Average amount of data passed on each IXGWRITE request during the interval.

This information is not included in the SMF record, it is calculated by the IXGPRT1 program by dividing SMF88LWB by SMF88LWI.

▶ **BYT Deleted interim ST w/o DASD (SMF88SIB)**

This is the number of bytes deleted from the interim storage, including the “hot air”, as a result of IXGDELET requests, where the data had not already been moved to an offload data set. For active log streams, this value should be close to SMF88SWB, meaning most data is being deleted before incurring the overhead of being moved to offload data sets.

For CF-Structure log streams, this value includes the amount of element space, rounded up to the element boundary for each IXGWRITE. The data is deleted from the CF structure without moving the data to an offload data set.

For DASD-only log streams, this value includes the amount of staging data set space rounded up to the CI boundary for each IXGWRITE. The data is deleted from the log stream staging data set without moving the data to an offload data set.

A non-zero value in this field indicates that IXGDELETs are being issued by the log stream connector.

► # Deletes w/o DASD write (SMF88SII)

The number of times a deletion from storage was performed where the data had not been moved to an offload data set.

This is another indication that IXGDELETs are being issued for this log stream.

► BYT Deleted interim ST w/DASD (SMF88SAB)

During the offload process, System Logger first physically deletes any logically deleted log blocks. If this is not sufficient to reduce the log stream to the LOWOFFLOAD value, it then moves log blocks, starting with the oldest one, to offload data sets until the LOWOFFLOAD threshold is reached.

This field contains the number of bytes which were deleted from interim storage during this interval after the data had been moved to offload data sets.

For CF-Structure log streams, this value includes the amount of “hot air” in each element.

For DASD-only log streams, this value includes the amount of “hot air” in the staging data set CIs. It is worth noting that System Logger does not read the staging data set when moving log blocks to the offload data set—it reads the local buffer (data space) copy and writes it to the offload data set. However, the SMF88SAB value represents the amount of space used in the staging data set since this resource is of limited size.

For both CF-Structure and DASD-only active log streams which do tail trimming with the expectation the data will be held within interim storage, this field should be zero for best performance.

► # Deletes w/write (SMF88SAI)

The number of times a deletion from storage was performed where the data had been moved to an offload data set. This is not affected by the RETPD specification: in this case data is offloaded, and then deleted.

For an active log stream, this is another reflection that IXGDELETs are not being issued frequently enough. For optimum performance this field should be zero.

► EVENT

– OFFLOAD (SMF88EO)

The number of offload processes which ran in the interval—note that this does *not* necessarily mean that any data was moved to the offload data sets.

– DASD SHFT (SMF88EDS)

The number of times an additional offload data set had to be allocated during the interval.

Monitor this field to see if the number of offload data sets being allocated is consistent with your requirements for this log stream. For critical log streams, this field should be zero or a low number during the prime processing period for best performance and availability.

– STRC FULL (SMF88ESF) (CF-Structure log streams only)

Number of times a structure-full condition was reached.

Following the application of OA03055, the System Logger increments SMF88ESF when a log stream exceeds its allowable limit in the structure. The structure may not actually be completely full.

Prior to APAR OA03055, SMF88ESF only reflects the number of times the structure was completely full rather than being log stream-based. The change provides a better correlation of SMF88ESF and Return Code 8 Reason Code 866 errors.

This field should be zero for best performance and availability.

– NTRY FULL (SMF88EFS) (CF-Structure log streams only)

Number of times an offload was initiated because the number of entries in use in the structure reached 90%. Note when this condition occurs, an offload is initiated for *all* log streams in the structure.

This field should be zero for best performance.

– STG THLD (SMF88ETT)

Number of times an IXGWRITE was issued and the staging data set had reached or exceeded its HIGHOFFLOAD threshold.

For a CF-Structure log stream, this value should be zero. If the value is greater than zero, it indicates the offload is being triggered because the staging data set hit its HIGHOFFLOAD threshold before the interim storage in the CF.—the size of the staging data set should be increased so it has the capacity to hold the same or a greater amount of data compared to the CF-Structure log stream storage.

For a DASD-only log stream, the value should be equal to or slightly higher than the number of offloads in the interval. A number greater than the number of offloads in the interval indicates a number of IXGWRITE calls were issued to the log stream while it was at or above the HIGHOFFLOAD threshold. A very large number indicates a possible slow down in the offload process.

– STG FULL (SMF88ETF)

The number of times the staging data set filled in the interval. This field applies to both CF-Structure and DASD-only log streams.

This field should be zero for best performance and availability. A non-zero value in this field commonly indicates one of the following:

- There is a problem with the offload process, meaning the System Logger could not move the data to the offload data set quickly enough.
- The staging data is too small for the volume of data being written to this log stream.
- The HIGHOFFLOAD threshold is too high, meaning the space between HIGHOFFLOAD and the data set being full is too small.

– RE-BLD (SMF88ERI) (CF-Structure log streams only)

The number of times during the interval the structure was rebuilt.

### 8.6.5 DFSORT jobs to format SMF 88 records

One of the disadvantages of the IXGRPT1 program is that it tries to squeeze so much information into 133 columns, resulting in its two-lines-per record format, making it difficult to read. Another disadvantage is that it processes the SMF 88 records by time rather than by log stream. So, you are given data for every log stream in one interval, then every log stream in the next interval, and so on. If you are trying to quickly review the activity for just one log stream, this can make it cumbersome.

Because of the format of some of the fields in the SMF record, it is necessary to use a DFSORT E15 exit to convert the data. The load module containing the exit, together with sample JCL to produce these reports are available as Additional Material for this book. For more information about this, refer to “Locating the Web material” on page 313. Sample of the report obtained through the exit are shown below.

The DFSORT job we used to solve these concerns, breaks the information contained in the IXGRPT1 report across three separate reports. Each report lists the information in log stream name, rather than time, sequence. This makes it easier to review the activity for a given log stream across the entire day.

The reports are broken down as follows:

- ▶ The first report, produced in step SHOWALL, simply lists every log stream for which Type 88 records were found, sorted by log stream name within system. It also lists how many Type 88 records were found for that log stream from that system. Figure 8-7 contains an excerpt from that report.

Logstream Names by System				
	10/15/03	22:23:18	- 1 -	
System ID	Logstream Name	Structure Name	Records	KB Written
ACPU	PROD.DBSF.DFHLOG	PROD_A002_LOG	98	46
ACPU	PROD.DBSF.DFHSHUNT	PROD_A002_SHUNT	98	0
ACPU	SYST.ALLE.DFHJ04	SYST_ALLE_J04	40	1383
ACPU	SYST.CSWBSTAT	SYST_CSWBSTAT	51	487
ACPU	SYST.DBAK.DFHLOG	SYST_ALLE_LOG	71	37
ACPU	SYST.DBAK.DFHSHUNT	SYST_ALLE_SHUNT	71	0
ACPU	SYST.DBAN.DFHLOG	SYST_ALLE_LOG	71	363
ACPU	SYST.DBAN.DFHSHUNT	SYST_ALLE_SHUNT	71	0
ACPU	SYST.DBAS.DFHLOG	SYST_ALLE_LOG	79	1072
ACPU	SYST.DBAS.DFHSHUNT	SYST_ALLE_SHUNT	79	0
ACPU	SYST.DBAU.DFHLOG	SYST_ALLE_LOG	71	32
ACPU	SYST.DBAU.DFHSHUNT	SYST_ALLE_SHUNT	71	0
ACPU	SYST.DBKA.DFHLOG	SYST_ALLE_LOG	72	688
ACPU	SYST.DBKA.DFHSHUNT	SYST_ALLE_SHUNT	72	0
ACPU	SYST.DBKB.DFHLOG	SYST_ALLE_LOG	71	7902
ACPU	SYST.DBKB.DFHSHUNT	SYST_ALLE_SHUNT	71	0

Figure 8-7 Sample report from step SHOWALL

- ▶ The next report, produced by step SHOWDEF1, contains information about the selected log stream(s) from a log stream perspective.

A sample of this report is shown in Figure 8-8. You will see that the column headings are the suffixes of the SMF 88 fields that the data comes from. For example, the first column after the time and date is the number of IXGWrites issued for the log stream in that interval (from SMF88LWI). The next two columns are the minimum (SMF88LIB) and maximum length (SMF88LAB) of the blocks passed on the IXGWrites. And so on. As you can see, all the fields in this report are reporting information about the usage of this log stream.

```

LOGG8R 88 LOGSTREAM SECTION      10/12/03      20:12:06      - 1 -
LOGSTREAM NAME  PROD.DBAP.DFHLOG

```

DATE	TIME	LWI/#	LIB/#	LAB/#	LWB/KB	LDB/KB	LIO/#	LIS/#
2003/06/30	00:00	0	120	7648	0	0	0	0
2003/06/30	00:17	3	120	7648	10	0	1	0
2003/06/30	01:30	4	120	443	1	0	0	0
2003/06/30	02:00	0	120	443	0	0	0	0
2003/06/30	02:30	1	120	443	0	0	0	0
2003/06/30	03:00	0	120	443	0	0	0	0
2003/06/30	03:30	0	120	443	0	0	0	0
2003/06/30	04:00	0	120	443	0	0	0	0
2003/06/30	04:30	0	120	443	0	0	0	0
2003/06/30	05:00	0	120	443	0	0	0	0
2003/06/30	05:30	2	120	4043	4	0	0	0
2003/06/30	06:00	0	120	4043	0	0	0	0
2003/06/30	06:30	0	120	4043	0	0	0	0
2003/06/30	07:00	0	120	4043	0	0	0	0
2003/06/30	07:30	6	120	7648	22	0	0	0
2003/06/30	08:00	16	120	7648	60	0	0	0
2003/06/30	08:30	20	120	7648	76	0	0	0
2003/06/30	09:00	90	120	7648	335	0	0	0
2003/06/30	09:30	98	120	7648	359	0	0	0
2003/06/30	10:00	92	120	7648	343	0	0	0
2003/06/30	10:30	62	120	7648	229	0	0	0
2003/06/30	11:00	70	120	7648	259	0	0	0
2003/06/30	11:30	74	120	7648	274	0	0	0
2003/06/30	12:00	56	120	7648	206	0	0	0
2003/06/30	12:30	24	120	7648	91	0	0	0
2003/06/30	13:00	40	120	7648	145	0	0	0

Figure 8-8 Sample log stream attribute report from step SHOWDEF1

- ▶ The next report, also produced by step SHOWDEF1, contains information about the log stream from an events perspective.

A sample of this report is shown in Figure 8-9 on page 300. While this report may appear very boring (nearly all the fields are zero), in this case, boring is good! A value of zero in most of these fields is an indicator of no problems (for example, no occurrences of DASD shifts (EDS), no occurrences of the staging data set hitting its threshold (ETT), and so on).

In this particular report, the one thing that should attract your attention is that there were so few (only 2) offloads (EO) initiated during the entire 16-hour interval. This is an indicator that the interim storage for this log stream is significantly larger than necessary.

```

LOGGER 88 EVENTS SECTION      10/12/03      20:12:07      - 1 -
LOGSTREAM NAME  PROD.DBAP.DFHLOG
DATE           TIME           EDS/#          ERI/#          ERC/#          ESF/#          ETT/#          ETF/#          EO/#          EFS/#          EDO/#
-----
2003/06/30    00:00             0              0              0              0              0              0              0              0              0
2003/06/30    00:17             0              0              0              0              0              0              1              0              0
2003/06/30    01:30             0              0              0              0              0              0              0              0              0
2003/06/30    02:00             0              0              0              0              0              0              0              0              0
2003/06/30    02:30             0              0              0              0              0              0              0              0              0
2003/06/30    03:00             0              0              0              0              0              0              0              0              0
2003/06/30    03:30             0              0              0              0              0              0              0              0              0
2003/06/30    04:00             0              0              0              0              0              0              0              0              0
2003/06/30    04:30             0              0              0              0              0              0              0              0              0
2003/06/30    05:00             0              0              0              0              0              0              0              0              0
2003/06/30    05:30             0              0              0              0              0              0              0              0              0
2003/06/30    06:00             0              0              0              0              0              0              0              0              0
2003/06/30    06:30             0              0              0              0              0              0              0              0              0
2003/06/30    07:00             0              0              0              0              0              0              0              0              0
2003/06/30    07:30             0              0              0              0              0              0              0              0              0
2003/06/30    08:00             0              0              0              0              0              0              0              0              0
2003/06/30    08:30             0              0              0              0              0              0              0              0              0
2003/06/30    09:00             0              0              0              0              0              0              0              0              0
2003/06/30    09:30             0              0              0              0              0              0              0              0              0
2003/06/30    10:00             0              0              0              0              0              0              0              0              0
2003/06/30    10:30             0              0              0              0              0              0              0              0              0
2003/06/30    11:00             0              0              0              0              0              0              0              0              0
2003/06/30    11:30             0              0              0              0              0              0              0              0              0
2003/06/30    12:00             0              0              0              0              0              0              0              0              0
2003/06/30    12:30             0              0              0              0              0              0              0              0              0
2003/06/30    13:00             0              0              0              0              0              0              0              0              0
2003/06/30    13:30             0              0              0              0              0              0              1              0              0
2003/06/30    14:00             0              0              0              0              0              0              0              0              0
2003/06/30    14:30             0              0              0              0              0              0              0              0              0
2003/06/30    15:00             0              0              0              0              0              0              0              0              0
2003/06/30    15:30             0              0              0              0              0              0              0              0              0
2003/06/30    16:00             0              0              0              0              0              0              0              0              0

```

Figure 8-9 Sample events report from step SHOWDEF1

- The third report produced by step SHOWDEF1, contains information about the log stream from the interim storage perspective.

A sample of this report is shown in Figure 8-11 on page 302. Amongst the interesting information this report provides is the amount of log data (in KB) written to interim storage for this log stream (SWB/KB), and the SC1, SC2, and SC3 fields which indicate the number of writes when interim storage was below the HIGHOFFLOAD threshold, over the threshold but below 90% full, and above 90% full.

```

LOGGER 88 STRUCTURE SECTION      10/12/03      20:12:07      - 1 -

LOGSTREAM NAME  PROD.DBAP.DFHLOG

  DATE      TIME      SWB/KB      SIB/KB      SAB/KB      SSI/#      SAI/#      SC1/#      SC2/#      SC3/#
-----
2003/06/30  00:00          0          0          0          0          0          0          0          0
2003/06/30  00:17         11         74          0         25          0          3          0          0
2003/06/30  01:30          1          0          0          0          0          4          0          0
2003/06/30  02:00          0          0          0          0          0          0          0          0
2003/06/30  02:30          0          0          0          0          0          1          0          0
2003/06/30  03:00          0          0          0          0          0          0          0          0
2003/06/30  03:30          0          0          0          0          0          0          0          0
2003/06/30  04:00          0          0          0          0          0          0          0          0
2003/06/30  04:30          0          0          0          0          0          0          0          0
2003/06/30  05:00          0          0          0          0          0          0          0          0
2003/06/30  05:30          4          0          0          0          0          2          0          0
2003/06/30  06:00          0          0          0          0          0          0          0          0
2003/06/30  06:30          0          0          0          0          0          0          0          0
2003/06/30  07:00          0          0          0          0          0          0          0          0
2003/06/30  07:30         23          0          0          0          0          6          0          0
2003/06/30  08:00         62          0          0          0          0         16          0          0
2003/06/30  08:30         77          0          0          0          0         20          0          0
2003/06/30  09:00        341          0          0          0          0         90          0          0
2003/06/30  09:30        365          0          0          0          0         98          0          0
2003/06/30  10:00        349          0          0          0          0         92          0          0
2003/06/30  10:30        233          0          0          0          0         62          0          0
2003/06/30  11:00        264          0          0          0          0         70          0          0
2003/06/30  11:30        280          0          0          0          0         74          0          0
2003/06/30  12:00        210          0          0          0          0         56          0          0
2003/06/30  12:30         93          0          0          0          0         24          0          0
2003/06/30  13:00        148        2029          0         550          0         38          2          0
2003/06/30  13:30        108          0          0          0          0         28          0          0
2003/06/30  14:00        140          0          0          0          0         38          0          0
2003/06/30  14:30        155          0          0          0          0         42          0          0
2003/06/30  15:00        217          0          0          0          0         56          0          0
2003/06/30  15:30        217          0          0          0          0         58          0          0
2003/06/30  16:00        140          0          0          0          0         38          0          0

```

Figure 8-10 Sample interim storage report from step SHOWDEF1

- ▶ The last report produced by step SHOWDEF1 pulls the information you are most likely to want to monitor together into a single report. A sample is shown in Figure 8-11. In this case, the columns headings are actually self-explanatory. This is a good report to use to get a feel for the activity of your critical log streams over the day.

```

Logger 88 Key Indicators      10/12/03      20:12:07      - 1 -
LOGSTREAM NAME  PROD.DBAP.DFHLOG
Date           Time           KB Deleted Interim ST W/DASD  DASD SHIFT  KB Written to Interim Storage  KB Deleted interim ST w/o DASD  KB Written by Users IXGWITES  STG THLC
-----
2003/06/30    00:00          0              0              0              0              0              0              C
2003/06/30    00:17          0              0              11             74             10             10             C
2003/06/30    01:30          0              0              1              0              1              1              C
2003/06/30    02:00          0              0              0              0              0              0              C
2003/06/30    02:30          0              0              0              0              0              0              C
2003/06/30    03:00          0              0              0              0              0              0              C
2003/06/30    03:30          0              0              0              0              0              0              C
2003/06/30    04:00          0              0              0              0              0              0              C
2003/06/30    04:30          0              0              0              0              0              0              C
2003/06/30    05:00          0              0              0              0              0              0              C
2003/06/30    05:30          0              0              4              0              4              4              C
2003/06/30    06:00          0              0              0              0              0              0              C
2003/06/30    06:30          0              0              0              0              0              0              C
2003/06/30    07:00          0              0              0              0              0              0              C
2003/06/30    07:30          0              0              23             0              22             22             C
2003/06/30    08:00          0              0              62             0              60             60             C
2003/06/30    08:30          0              0              77             0              76             76             C
2003/06/30    09:00          0              0              341            0              335            335            C
2003/06/30    09:30          0              0              365            0              359            359            C
2003/06/30    10:00          0              0              349            0              343            343            C
2003/06/30    10:30          0              0              233            0              229            229            C
2003/06/30    11:00          0              0              264            0              259            259            C
2003/06/30    11:30          0              0              280            0              274            274            C
2003/06/30    12:00          0              0              210            0              206            206            C
2003/06/30    12:30          0              0              93             0              91             91             C
2003/06/30    13:00          0              0              148            2029           145            145            C
2003/06/30    13:30          0              0              108            0              106            106            C
2003/06/30    14:00          0              0              140            0              137            137            C
2003/06/30    14:30          0              0              155            0              152            152            C
2003/06/30    15:00          0              0              217            0              213            213            C
2003/06/30    15:30          0              0              217            0              213            213            C
2003/06/30    16:00          0              0              140            0              137            137            C

```

Figure 8-11 Sample key indicator report from step SHOWDEF1

## 8.6.6 Sample spreadsheet tool to analyze SMF 88 records

Finally, to help you identify log streams that warrant attention with minimal effort, we have developed a spreadsheet-based reporting tool. The tool accepts SMF Type 88 records as input, loads them into a small set of DB2 tables on MVS, then loads the spreadsheet using an SQL query over FTP from the workstation.

The resulting spreadsheet contains the following worksheets, providing various views of all the data in the records, together with a set of exception reports. The exception reports are based on the tuning guidance provided in this document. The complete set of sheets is as follows:

- ▶ Overview - Selected Interval
- ▶ Overview - LogStream Counters
- ▶ Overview - LogStream Counters
- ▶ Overview - LogStream Counters
- ▶ Overview - System Activity Summary
- ▶ Overview - System Activity Details by Interval
- ▶ Overview - System Activity Summary by Logstream
- ▶ Overview - Logstream Activity Summary
- ▶ Overview - Logstream Activity Details - Selected Logstream
- ▶ Overview - Logstream Activity Details - All the Logstreams
- ▶ Overview - Logstream Connectors - All the Logstreams by Interval



- ▶ Overview - LogStream Blocksize and Hot Air factors
- ▶ Exceptions - SMF88ESF GT 0 Summary
- ▶ Exceptions - SMF88ESF GT 0 Details
- ▶ Exceptions - SMF88ETF GT 0 Summary
- ▶ Exceptions - SMF88ETF GT 0 Details
- ▶ Exceptions - SMF88EFS GT 0 Summary
- ▶ Exceptions - SMF88EFS GT 0 Details
- ▶ Exceptions - SMF88SC3 GT 0 Summary
- ▶ Exceptions - SMF88SC3 GT 0 Details
- ▶ Exceptions - SMF88SAB GT 0 Summary
- ▶ Exceptions - SMF88SAB GT 0 Details
- ▶ Exceptions - SMF88SIB EQ 0 Summary
- ▶ Exceptions - SMF88SIB EQ 0 Details
- ▶ Exceptions - SMF88EO GT 0 Summary
- ▶ Exceptions - SMF88EO GT 0 Details
- ▶ Exceptions - SMF88EDS GT 0 Summary
- ▶ Exceptions - SMF88EDS GT 0 Details
- ▶ Exceptions - SMF88ERI GT 0 Summary
- ▶ Exceptions - SMF88ERI GT 0 Details

Generally speaking, the Exceptions sheets should all be empty. However, for funnel-type log streams (such as OPERLOG or IMS Shared Message Queue), it would be normal to see entries in the SMF88SIB and SMF88EDS sheets; for active log streams, these should also be empty.

The spreadsheet tool, together with instructions on its use, is included in the Additional Material for this book. For more information about this, see “Using the Web material” on page 313.

Archived



## Disaster Recovery considerations

This chapter describes the special considerations for System Logger and log streams in a disaster recovery environment. Details are provided about:

- ▶ How to provide time consistent data
- ▶ PPRC and System Logger
- ▶ XRC and System Logger

## 9.1 Overview

The System Logger environment consists of several components, and the data across all the components must be in a time consistent state for it to be used during a disaster recovery situation. These pieces include:

- ▶ The System Logger policy (which records the list of offload data sets associated with each log stream along with the highest used RBA in the current data set)
- ▶ The catalog structure (which maintains the location of all the System Logger data sets)
- ▶ The System Logger coupling facility structures

If you dump the volume with the master catalog (or user catalog) on it and then dump the volume with the System Logger policy on it, then the policy might indicate that certain data sets should exist, but those data sets might not be reflected in the catalog.

If your goal is to preserve the System Logger environment without losing any data, then you cannot rely on volume dumps to transport the data to the recovery site.

## 9.2 Providing time consistent data

Since you cannot rely on volume dumps to restore the System Logger environment at a recovery site, you must use some mechanism that provides for time consistent data across all the components. However, simply mirroring the data from one site to another is not sufficient as you are still exposed to the possible inconsistency of data across multiple control units.

One option for providing time consistent data is IBM GDPS service offering, and there are two variations of GDPS: GDPS/PPRC and GDPS/XRC. Using either of these products allows the disk mirroring of staging data sets, the System Logger policy (couple data set), the MVS catalogs, all System Logger exploiters' data and all the System Logger data sets to consistent state to the disaster recovery site. Only when the secondary site has the data in a consistent state can the exploiters use their log streams in the recovery site.

### 9.2.1 Peer to Peer Remote Copy - PPRC

PPRC is a synchronous data mirroring technology: data is committed to both primary and secondary disk subsystems before the write is confirmed to be complete. As a consequence, this technology is sensitive to performance impact as the distance between primary and secondary disk subsystems increases.

At least one system in the recovery site is a member of the same parallel sysplex with the systems in the primary site. The maximum distance between the two sites is 40 kilometers, and this limiting factor is imposed by the Sysplex timer links.

With a synchronous mirroring technology secondary data consistency does not come automatically: it needs to be managed. When GDPS/PPRC detects a freeze trigger (an indication that the disk mirroring is inoperative), it issues the PPRC commands to freeze all the disk mirroring to the secondary disk subsystems. As a result of the freeze, all the data on disk is time consistent.

To support System Logger in a GDPS/PPRC environment, you must:

- ▶ Ensure that all CF-Structure log streams are defined to unconditionally use staging data sets and the volumes that house the staging data sets must be mirrored to the recovery site.
- ▶ The System Logger policy primary Couple Data Set must reside on a volume that is being mirrored to the recovery site

Unfortunately, including the System Logger Couple Data Set on volumes in the PPRC configuration makes the recovery a little more complicated.

See Appendix A in *GDPS/PPRC V2R8 Installation and Customization Guide*, ZG24-6703, for more details.

## 9.2.2 Extended Remote Copy - XRC

GDPS also supports asynchronous disk mirroring technology with the GDPS/XRC version of the offering. With GDPS/XRC, there is not distance limitation on the distance between the two sites as the systems in the two sites are not part of the same sysplex.

Since GDPS/XRC does not impose any distance limitations, it can provide coverage for more regional type disasters such as earthquakes, blizzards, hurricanes and flooding.

Propagation delays resulting from long distances mean that in many cases synchronous data mirroring would have too much of a performance impact. That's the main reason for the existence of extended Remote Copy (XRC), a software centric data mirroring solution which operates asynchronously: updating of secondary devices is not time-coordinated with the updating of the primary devices.

It is important to remember that since the copy technology is asynchronous, some data loss is to be expected in a failover.

In this environment, you must again ensure that all production disk is included in the mirroring configuration.

See *GDPS/XRC V2R8 Installation and Customization Guide*, ZG24-6704, for more details on GDPS/XRC.

Archived



## Rebuilding the LOGR Policy

This appendix provides a sample of a REXX program that can be used to rebuild the LOGR policy using as input the report obtained with the IXCMIAPU utility with the REPORT(YES) keyword.

## JCL used to rebuild the LOGR policy

The skeleton in Example A-1 is a JCL sample used to rebuild the input statements for the LOGR policy. There are two steps in the job:

1. The first step is a LIST of the current LOGR policy content where the output is put into a temporary sequential data set.
2. The second step is the invocation of the LOGRPOL REXX CLIST to read the output report from the temporary data set and rebuild from them the LOGR policy definition into a new data set.

*Example: A-1 JCL Skeleton sample*

---

```
//JLOGRPOL JOB (0,0), 'JLOGRPOL', CLASS=A, MSGCLASS=X, NOTIFY=&SYSUID
//*****
/* THIS JOB BUILDS IXCMIAPU STATEMENTS TO DEFINE THE LOGGER-RELATED
/* STRUCTURES AND LOGSTREAMS.
/* IT IS VERY USEFUL IF THE JOB USED TO MAINTAIN THE LOGGER
/* DEFINITIONS GETS LOST!
//*****
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD DISP=(,PASS), DSN=&&LOGR, SPACE=(TRK,(5,5))
//SYSIN DD *
    DATA TYPE(LOGR) REPORT(NO)
    LIST STRUCTURE NAME(*)
    LIST LOGSTREAM NAME(*)
/*
//STEP2 EXEC PGM=IKJEFT01, PARM='LOGRPOL'
//SYSTSIN DD DUMMY
//SYSTSPT DD SYSOUT=*
//SYSEXEC DD DISP=SHR, DSN=ESA.SYS1.CLIST
//LOGRI DD DISP=(OLD,DELETE), DSN=&&LOGR
//LOGRO DD DSN=ESA.SYS1.JCL(LOGRPOLN), DISP=SHR
```

---

Here is the content of the LOGRPOLN member to show how the LOGR statement are built by the LOGRPOL CLIST (Example A-2).

*Example: A-2 LOGR Policy statements*

---

```
DEFINE STRUCTURE
    NAME(CIC_DFHLOG_001)
    LOGSNUM(20)
    MAXBUFSIZE(64000)
    AVGBUFSIZE(640)
DEFINE STRUCTURE
    NAME(CIC_DFHSHUNT_001)
    LOGSNUM(20)
    MAXBUFSIZE(64000)
    AVGBUFSIZE(640)
DEFINE LOGSTREAM
    NAME(@@C.#@$CCM$2.DFHLOG2)
    STRUCTNAME(CIC_DFHLOG_001)
    LS_DATACLAS(LOGR24K)
    HLQ(IXGLOGR)
    MODEL(NO)
    LS_SIZE(16384)
    STG_SIZE(0)
    LOWOFFLOAD(0)
    HIGHOFFLOAD(80)
    STG_DUPLEX(NO)
```



```

        RETPD(0)
        AUTODELETE(NO)
        OFFLOADRECALL(YES)
        DASDONLY(NO)
        DIAG(NO)
        LOGGERDUPLEX(UNCOND)
        EHLQ(NO_EHLQ)
DEFINE LOGSTREAM
        NAME(#@%C.#@%CCM$1.DFHSUN2)
        STRUCTNAME(CIC_DFHSUNT_001)
        LS_DATACLAS(LOGR24K)
        HLQ(IXGLOGR)
        MODEL(NO)
        LS_SIZE(16384)
        STG_SIZE(0)
        LOWOFFLOAD(0)
        HIGHOFFLOAD(80)
        STG_DUPLEX(NO)
        RETPD(0)
        AUTODELETE(NO)
        OFFLOADRECALL(YES)
        DASDONLY(NO)
        DIAG(NO)
        LOGGERDUPLEX(UNCOND)
        EHLQ(NO_EHLQ)

```

---

## LOGRPOL CLIST

Example A-3 is the CLIST used in the JCL to format the IXCMIAPU output and rebuild the input statement for a LOGR policy.

*Example: A-3 LOGRPOL REXX Clist*

---

```

/* rexx */
if sysvar("SYSENV") = "FORE" then do
  address TSO "ALLOC DD(LOGRI) DS(LOGR.POLICY) OLD"
  address TSO "ALLOC DD(LOGRO) DS(LOGR.SYSIN) MOD LRECL(80) RECFM(F B)"
end
"EXECIO * DISKR LOGRI (stem logri. finis"
done = 0
logro = 0
do I = 1 to logri.0 while ~done
  j = 0
  parse var logri.i 2 rest
  if rest = "LOGR Inventory Record Summary:" then done = 1
  do until rest = ""
    j = j + 1
    parse var rest word.j rest
    left_b = pos("(",word.j)
    right_b = pos(")",word.j)
  if (left_b+1) = right_b then word.j = ""
  if ~good_word(word.j) then word.j = ""
  select
    when word.j = "LOGSTREAM" then call write_out "DEFINE" word.j
    when word.j = "STRUCTURE" then call write_out "DEFINE" word.j
    when word.j ~= "" then call write_out "      " word.j
    otherwise nop
  end
end

```

```

    end
  end
  "EXECIO" logro "DISKW LOGRO (FINIS"
  exit
good_word:
  parse arg test_string
  if pos(",",test_string) = 0 then parse upper var test_string test_word
  else parse upper var test_string test_word "(" test_value ")" .
  select
    when test_word = "LOGSTREAM" then return 1
    when test_word = "NAME"      then return 1
  when test_word = "STRUCTNAME" then return 1
    when test_word = "LS_DATACLAS" then return 1
    when test_word = "LS_MGMTCLAS" then return 1
    when test_word = "LS_STORCLAS" then return 1
    when test_word = "HLQ"         then return 1
    when test_word = "MODEL"       then return 1
    when test_word = "LS_SIZE"     then return 1
    when test_word = "STG_MGMTCLAS" then return 1
    when test_word = "STG_STORCLAS" then return 1
    when test_word = "STG_DATACLAS" then return 1
    when test_word = "STG_SIZE"   then return 1
    when test_word = "LOWOFFLOAD" then return 1
    when test_word = "HIGHOFFLOAD" then return 1
    when test_word = "STG_DUPLEX" then return 1
    when test_word = "DUPLEXMODE" then return 1
    when test_word = "RMNAME"     then return 1
    when test_word = "DESCRIPTION" then return 1
    when test_word = "RETPD"      then return 1
    when test_word = "AUTODELETE" then return 1
  when test_word = "OFFLOADRECALL" then return 1
    when test_word = "DASDONLY"   then return 1
    when test_word = "DIAG"       then return 1
    when test_word = "LOGGERDUPLEX" then return 1
    when test_word = "EHLQ"      then return 1
    when test_word = "STRUCTURE"  then return 1
    when test_word = "LOGSNUM"    then return 1
    when test_word = "MAXBUFSIZE" then return 1
    when test_word = "AVGBUFSIZE" then return 1
    otherwise return 0
  end

write_out: procedure expose logro
  parse arg write_text
  logro = logro + 1
  queue write_text
  return 0

```

---

## Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

### Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246898>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the book form number, SG246898.

### Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
<b>SG246898.zip</b>	Zipped Code, Instructions and readme file for DFSORT tool
<b>SMF0881 for DB2 V1R5M0.zip</b>	Zip file containing readme file, JCL, spreadsheet macro, and tuning presentation for SMF88 spreadsheet tool

### How to use the Web material

Once you download the file, unzip it and open the readme file. This file contains the description of the other files and the instructions about how to upload and use the files.

Archived

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 316. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *CICS and VSAM Record Level Sharing: Recovery Considerations*, SG24-4768
- ▶ *Coupling Facility Structure Management and IMS*, TIPS-0113
- ▶ *DFSMSStvs Overview and Planning Guide*, SG24-6971
- ▶ *IBM Tools: CICS Performance Analyzer V1.2*, SG24-6882
- ▶ *IMS/ESA Shared Queues: A Planning Guide*, SG24-5257
- ▶ *IMS/ESA Version 6 Guide*, SG24-2228
- ▶ *IMS/ESA Version 6 Shared Queues*, SG24-5088
- ▶ *Merging Systems into a Sysplex*, SG24-6818

## Other publications

These publications are also relevant as further information sources:

- ▶ *CICS Transaction Server for z/OS Installation Guide*, GC34-5985
- ▶ *CICS Transaction Server for z/OS Release Guide*, GC34-5983
- ▶ *CICS Transaction Server for z/OS System Definition Guide*, GC34-5988
- ▶ *IMS Common Queue Server Guide and Reference*, SC27-1292
- ▶ *z/OS MVS Assembler Services Guide*, SA22-7605
- ▶ *z/OS MVS Authorized Assembler Services Reference ENF-IXG*, SA22-7610
- ▶ *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589
- ▶ *z/OS MVS Diagnosis: Reference*, GA22-7588
- ▶ *z/OS MVS Planning: APPC/MVS Management*, SA22-7599
- ▶ *z/OS MVS Planning: Operations*, SA22-7601
- ▶ *z/OS MVS Setting Up a Sysplex*, SA22-7625
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- ▶ *z/OS DFSMS: Using Data Sets*, SC26-7410
- ▶ *z/OS DFSMS: Implementing System-Managed Storage*, SC26-7407
- ▶ *z/OS MVS Planning: Workload Management*, SA22-7602
- ▶ *z/OS Sysplex Services Guide*, SA22-7617
- ▶ *z/OS System Management Facilities (SMF)*, SA22-7630

- ▶ *z/OS System Commands*, SA22-7627
- ▶ *DFSMStvs Planning and Operating Guide*, SC26-7348
- ▶ *DFSMStvs Administration Guide*, GC26-7483
- ▶ *GDPS/XRC V2R8 Installation and Customization Guide*, ZG24-6704
- ▶ *GDPS/PPRC V2R8 Installation and Customization Guide*, ZG24-6703

## Online resources

These Web sites are also relevant as further information sources:

- ▶ The CFSizer should be used to obtain accurate sizes for CF structures  
<http://www.ibm.com/servers/eserver/zseries/cfsizer/>
- ▶ CICS Servers  
<http://www.ibm.com/software/ts/cics/txppacs/cs1e.html>
- ▶ Parallel Sysplex  
<http://www.ibm.com/servers/eserver/zseries/pso>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

# Index

## Symbols

//SYSIN DD \* DATA TYPE(CFRM) STRUCTU 223

## A

- activity keypoint
  - how this affect logger in TVS 91
- activity keypoint frequency 91
- AKPFREQ 187, 195, 203
  - Sample report 203, 205
- ALLOWAUTOALT
  - CICS log stream 156
  - Transactional VSAM log stream 94
- ALLOWAUTOALT CFRM parameter 31
- APPC/MVS
  - adding log stream after IPL 215
  - CF failure 215
  - CF versus DASD-only log stream 210
  - CFRM definitions 212
  - criticality of data 210–211
  - enq problem 214
  - log stream description 210
  - log stream name definition 212
  - log stream sizing 211
  - logger definitions 212
  - losing data 211
  - performance considerations 215
  - security definitions 214
- AUTODELETE 172
  - APPC/MVS log stream 210, 214
  - CICS log stream 159, 164
  - LOGREC log stream 219
  - OPERLOG log stream 226
  - RRS log stream 236
  - Transactional VSAM log stream 97, 101
  - WebSphere for z/OS log stream 250
- AUTODELETE attribute
  - impact on when data is deleted from log stream 64
- AVGBUFSIZE
  - APPC/MVS log stream 213
  - definition 33
  - in relation to entry to element ratio 26
  - LOGREC log stream 218
  - OPERLOG log stream 224
  - Transactional VSAM log stream 96

## B

- BBOERRLG 249
- BBORBLOG 251
- BLOCKID
  - for log blocks 18
  - use when deleting log blocks 21

## C

- CF
  - volatile storage 198
- CF structures
  - APPC/MVS log stream failure 215
  - constraint in number 231
  - defining in CFRM policy 30
  - defining in System Logger policy 32
  - deleting 58
  - impact of size on offload processing 63
  - importance of CF volatility state 43, 68–69, 74
  - peer recovery 71
  - rebuild processing 73
  - specifying structure size 31
  - storage usage 26
- CFRM definitions 218
- CF-Structure versus DASD-only
  - performance considerations 282
- CICS
  - activity keypoint 149
  - AKPFREQ 186
  - Auto journal 150
  - Auto Journal sizing 183
  - CFCC level consideration 176
  - chaining of log records 146, 149
  - changing log stream names 147
  - CICS catalog and log streams 147
  - deleting data from DFHLOG log stream 146
  - DFHLOG sizing 177
  - DFHLSCU utility 186
  - DFHSHUNT sizing 177
  - forward recovery log 149
  - Forward Recovery sizing 184
  - JOURNALMODEL
    - identifying associated log streams 146
  - log of logs 150
  - log stream model 153
  - log stream sizing 175
  - non-system log stream sizing 180
  - overriding default log stream model names 147
  - overview of System Logger use 146
  - shutdown assist program 148
  - syncpoint processing 146
  - system log
    - log streams 149
    - system log sizing formula 179
    - types of log streams used by CICS 149
  - UOW definition 146
  - user journal log streams 146
  - user journals 150
    - shutdown type considerations 148
- CICS Performance Analyzer 188, 193
- CICS/VR 86–87, 150
- commands to switch LOGR CDS 14
- COUPLExx member 10, 14

Coupling facility  
volatility 282

## D

D GRS command 258  
D LOGGER command 106  
    examples 254  
    use in problem diagnosis 78  
D LOGREC command 220  
D SMS,LOG command 106  
D SMS,TRANVSAM command 106  
DASDONLY  
    Transactional VSAM log stream 101  
DASD-only log streams  
    deciding when to use 24  
    defining in System Logger policy 46  
    deleting 58  
    description 45  
    duplexing 54  
    HIGHOFFLOAD 281  
    how BLOCKIDs are generated 18  
    LOWOFFLOAD 281  
    migrating to CF-based log streams 56  
    offload processing 59, 64  
    recovery 72  
    recovery considerations 72  
    setup tasks 10  
    staging data set naming conventions 9  
deleting data from a log stream 20, 38, 50  
deleting log streams 57  
DFH0STAT 188  
    LGDFINT 190  
    MAXBUFSIZE 189  
DFHCESD shutdown assist program 148  
DFHLOG  
    criticality of data 150  
    default names for model log streams 147  
    deleting data from the log stream 149  
    description 149  
    determining the history point 149  
    migration consideration 173  
    when CICS connects 148  
DFHLSCU 177  
DFHSHUNT  
    criticality of data 150  
    default names for model log streams 147  
    deleting data from the log stream 149  
    description 149  
    migration consideration 173  
    when CICS connects 148  
    when records get written to this log stream 149  
DFHSTUP 188, 192  
DFSORT 188  
DIAG  
    CICS log stream 160, 165  
    Transactional VSAM log stream 97, 101  
DIAG log stream attribute 42, 54  
displaying LOGR CDS format values 14  
DSEXTENT 13  
    how many to specify 13

DSEXTENT IXCL1DSU keyword 13  
DUMP command 270  
duplexing  
    DASD-only log stream 283  
duplexing interim storage data  
    for CF-based log streams 42  
    for DASD-only log streams 54  
    single point of failure considerations 67  
DUPLEXMODE 282  
    APPC/MVS log stream 213  
    COND 282  
    OPERLOG log stream 225  
    Transactional VSAM log stream 99  
    UNCOND 282  
DUPLEXMODE attribute  
    single point of failure considerations 67  
DUPLEXMODE keyword 34

## E

element sizes 26  
entries and elements 26  
entry-to-element ratio  
    calculating 26  
    determining the current one for a structure 27  
    dynamic adjustments 27  
EREP utility 220

## F

failed-persistent connections  
    recovering 70  
FIRSTHIT 161, 245  
FULLTHRESHOLD  
    CICS log stream 155  
    Transactional VSAM log stream 94

## G

GDPS 306–307  
    PPRC solution 306  
    XRC solution 307  
GRSRNLxx inclusion list 10

## H

HIGHOFFLOAD  
    APPC/MVS log stream 213  
    CICS log stream 160, 165  
    OPERLOG log stream 224  
    RRS log stream 237  
    setting the threshold 285  
    Transactional VSAM log stream 97, 102  
    WebSphere for z/OS log stream 250  
HLQ  
    APPC/MVS log stream 214  
    LOGREC log stream 219  
    OPERLOG log stream 225

## I

IEADMC parmlib member 270



- IEAMDBLG 222, 228
- IGDSMS 90
- IGWSHUNT.SHUNTLOG CFRM definition 93
- INDOUBT UOWs in CICS 149
- INITSIZE
  - CICS log stream 156
  - Transactional VSAMlog stream 93
- interim storage
  - definition 8
  - duplexing 281
  - sizing 275
- IXCL1DSU
  - DSEXTENT keyword 13
  - LSR keyword 12
  - LSTRR keyword 12
  - SMDUPLEX keyword 14
- IXCL1DSU utility 12
- IXCMIAPU utility 15, 30, 32–33, 46, 55, 57
  - defining model log streams 147
  - LIST LOGSTREAM option 256
  - LIST option 256
  - LIST STRUCTURE option 256
  - required SAF definitions 16
  - role in problem determination 79
- IXGBRWSE service 19
- IXGCONN service 17
- IXGDELET service 20, 64
- IXGINVNT service 16, 147
- IXGRPRT1
  - OFFLOAD 196
- IXGRPT1 188, 195, 291
  - AKPFREQ 195
  - description 271
  - observations 195
  - provided informations 271
  - sort sample job 297
- IXGWRITE service 18

## J

- JOURNALMODEL 151
- JOURNALMODEL, using to define log streams 146
- JOURNALNAME 150, 152

## L

- LGDFINT 190
- locking
  - considerations in a CICS environment 84
- log block
  - definition 10
  - how they are processed by System Logger 19
- log record
  - definition 9
- log streams
  - attributes of CF-based ones 23
  - attributes that can be dynamically updated 55
  - characteristics 197
  - CICS log stream sizing 181
  - controlling duplexing of interim storage data 34
  - controlling offload processing 40, 52

- converting log stream from DASDONLY to CF-based 56
- DASD-only
  - description 45
- deciding between DASD-only and CF-based ones 24
- deciding between DASD-only or CF-based ones 22
- defining a model log stream 40, 52
- defining in System Logger policy 33
- definition 8
- deleting 57
- deleting data from the log stream 64, 146
- deletion 265
- deletion with active internal connection 265
- dynamic definition by CICS 146
- how many to have in a structure 25, 32
- loss of data 259
- MAXBUFSIZE 197
- model log streams
  - naming required by CICS 147
  - overriding default CICS names 147
  - relation to JOURNALMODEL in CICS 147
  - use by CICS 146
- offload data set names 33, 46
- operator initiated rebuild 266
- placement 197
- planning for peer recovery 25
- rebuild after a connectivity failure 269
- rebuild after a failure 268
- rebuilding 266
- single point of failure considerations 34, 54, 67–69, 72
- specifying retention periods for log data 38, 50
- staging data set names 33, 46
- temporary write errors 73
- types used by CICS 149
- using a model log stream 41, 53
- which ones should reside in the same structure 25, 27
- log tail management 149
- logger address space
  - starting 262
  - stopping 262
- LOGGERDUPLEX
  - CICS log stream 162
  - Transactional VSAM log stream 98
- LOGR CDS
  - definition 9
  - directory extent full condition 262
  - DSEXTENT formatting keyword 13
  - format level 13–14
    - displaying 14
  - format level considerations 14, 55
  - formatting considerations 12
  - LSR formatting keyword 12
  - LSTRR formatting keyword 12
  - printing contents for problem determination 79
  - procedure to moving to new one 14
  - requirements 12
  - switch of couple data sets 263
- LOGREC 218

- criticality of data 216
- how to prevent a loss of data 221
- log stream description 216
- log stream naming convention 216
- log stream sizing 217
- logger definitions 218
- migration considerations 217
- parmlib definition 217
- performance considerations 221
- security definitions 219

#### LOGSNUM 285

- APPC/MVS log stream 213
- OPERLOG log stream 224
- specifying maximum number of log streams in a structure 32
- Transactional VSAM log stream 96

#### LOGSTREAMID 86

#### LOWOFFLOAD

- APPC/MVS log stream 213
- CICS log stream 160, 165
- LOGREC log stream 218
- OPERLOG log stream 225
- RRS log stream 237
- setting the threshold 284
- Transactional VSAM log stream 102
- WebSphere for z/OS log stream 250

#### LS\_DATACLAS

- APPC/MVS log stream 213
- CICS log stream 161, 166
- OPERLOG log stream 225
- RRS log stream 237
- Transactional VSAM log stream 98, 102

#### LS\_SIZE

- APPC/MVS log stream 213
- CICS log stream 160, 165
- LOGREC log stream 219
- OPERLOG log stream 225
- RRS log stream 237
- Transactional VSAM log stream 98, 102
- WebSphere for z/OS log stream 250

#### LSR IXCL1DSU keyword 12

#### LSTRR IXCL1DSU keyword 12

## M

#### MAXBUFSIZE 285

- APPC/MVS log stream 213
- CICS log stream 157, 167
- definition 32
- in relation to entry to element ratio 26
- LOGREC log stream 218
- RRS log stream 234, 240
- Transactional VSAM log stream 96
- WebSphere for z/OS log stream 249

#### MINSIZE

- CICS log stream 156

#### model log streams

- use by CICS 146–147

#### MVSADMIN.LOGR SAF FACILITY class 16

#### MVSADMIN.XCF.CFRM SAF FACILITY class 16

## O

#### offload data sets

- allocation 60
- definition 9
- DSEXTENT requirement 13
- impact of errors during data set deletion processing 66
- migration considerations 264
- naming convention 9
- naming conventions 171
- orphans 66, 257
- pending 258
- recommended CI Size 49
- SHAREOPTIONS requirement 11
- sizing 274
- sizing considerations for deletion 66
- SMS data class 36, 48
- specifying data set sizes 38, 50
- specifying HLQ 39, 51
- when are they deleted 65

#### offload processing 58, 62, 275

- controlling whether migrated data sets are recalled 42, 54
- definition 9
- deleting offload data sets 65
- monitoring 261
- recommended threshold values 62
- specifying high offload threshold 40, 52
- specifying low offload threshold 40, 52

#### OFFLOADRECALL

- CICS log stream 163, 167
- Transactional VSAM log stream 99, 103

#### OPERLOG

- CFRM definitions 223
- criticality of data 222
- disabling 226
- enabling 226
- how to calculate AVGBUFSIZE 224
- log stream description 222
- log stream sizing 222
- logger definition 223
- managing the log data 222
- parmlib definition 223
- performance considerations 228
- relationship with SYSLOG 228
- SDSF considerations 227
- security definitions 226

## P

#### peer recovery 71

- requirements 71

#### PPRC 306

#### PREFLIST

- Transactional VSAM log stream 94

#### problem determination

- gathering required documentation 42, 54, 77
- use of DIAG(YES) 42, 79

## R

### RACF

- accesses required to be able to use log streams 21
- accesses required to update System Logger policy 16
- resources associated with System Logger 16

### rebuild

- after a CF failure 268
- after a connectivity failure 269
- operator request 266

### recommendations

- configure to allow for peer recovery 71
- high offload threshold 63
- how to delete a log stream with valid data in interim storage 265
- MAXBUFSIZE value 33
- never delete offload data sets manually 66
- not to use ALLOWAUTOALTER 31
- offload thresholds 62
- recommended APARs 69
- service strategy for System Logger 77
- System Logger maintenance policy 80
- to always specify an LS\_SIZE value 38, 50
- to define staging data set attributes for all log streams 34
- to use caution with AUTODELETE parameter 38, 50
- to use CI Size of 24K for offload data sets 37, 49
- use DFHCESD to protect user journal log records 148
- use of DIAG(YES) 42, 54
- use of OFFLOADRECALL(NO) 42, 54
- use XLGSTRM exit to tailor CICS log stream attributes 147

### Record Level Sharing

- description 83

### Redbooks Web site 316

- Contact us xiii

### Report

- CICS Sample 198
- LGDFINT sample 200–202

### RESERVE

- affecting performance 286

### RESOURCE(IXLSTR.structurename) SAF FACILITY

class 16

### RESOURCE(log\_stream\_name) SAF LOGSTRM class

16

### RETPD 172

- APPC/MVS log stream 210, 214
- CICS log stream 159, 164
- LOGREC log stream 219
- OPERLOG log stream 226
- RRS log stream 236
- Transactional VSAM log stream 97, 101
- WebSphere for z/OS log stream 250

### RETPD attribute

- impact on when data is deleted from log stream 64
- impact on when offload data sets are deleted 66

### RMF 188, 287

- Coupling Facility Activity Report 287
- Workload Activity Report 289

## RRS

- CF vs DASD-only log stream 228
- CFRM definitions 232
- criticality of data 230
- DASD-only log stream definitions 238
- log stream description 228
- log stream naming conventions 231
- log stream sizing 230
- logger definitions 233
- logging group 229
- managing log data 231
- performance considerations 242
- power-on-reset considerations 241
- security definitions 240
- shutdown 241
- start up 241
- start up cold 242
- syncpoint manager for TVS 84

## S

### SETLOGRC command 216–217, 220

### SETRRS command 241

### SHAREOPTIONS

- use for offload data sets 66
- use of with staging and offload data sets 47–48

### SIZE

- CICS log stream 156–157
- Transactional VSAM log stream 93

### sizing

- APPC/MVS log stream 211
- duplex copy of log stream 283
- interim storage 275
- interim storage for active log stream 277
- interim storage for funnel-type 275
- Logrec log stream 217
- offload data sets 274
- OPERLOG log stream 222
- RRS log stream 230
- Web-based tool 217
- web-based tool 231
- WebSphere for z/OS log stream 246

### sizing DASD-only for active log stream 280

### sizing DASD-only funnel-type 278

### SMDUPLEX IXCL1DSU keyword 13

### SMF 88 291

### SMF Type 88 records

- description 270
- introduction 77
- limitations 77
- subtypes 77
- when they are produced 77

### SMS subsystem

- System Logger requirements 11

### SMSVSAM 83

### staging data sets

- definition 9
- naming convention 9
- naming conventions 171
- required CI Size 35, 47
- SHAREOPTIONS requirement 11

- sizing for CF log streams 63
- sizing for DASD-only log streams 64
- SMS data class 47
- specifying data set sizes 36, 48
- specifying HLQ 39, 51
- use if CF structure fails 73
  - when they are allocated 18
- starting the System Logger subsystem 69
- STG\_DATACLAS
  - APPC/MVS log stream 213
  - RRS log stream 238
  - Transactional VSAM log stream 99, 103
- STG\_DUPLEX
  - APPC/MVS log stream 213
  - CICS log stream 162, 167
  - OPERLOG log stream 225
  - RRS log stream 237, 240
  - Transactional VSAM log stream 99
- STG\_DUPLEX attribute
  - role during volatility changes 74
  - single point of failure considerations 67, 69
  - when it can be ignored 73
- STG\_DUPLEX keyword 34
- STG\_SIZE
  - APPC/MVS log stream 213
  - CICS log stream 162, 166
  - RRS log stream 237
  - Transactional VSAM log stream 100, 103
- stopping the System Logger subsystem 69
- STRUCTNAME
  - Transactional VSAM log stream 100
- structures
  - deleting from System Logger policy 58
  - how storage is allocated to log streams 25
- System Automation for OS/390
  - CF versus DASD-only 243
  - CFRM definitions 243
  - criticality of data 243
  - data gap in log stream 245
  - log stream description 242
  - log stream naming conventions 243
  - log stream sizing 243
  - logger definitions 244
  - performance considerations 246
  - security considerations 245
  - start up problem 245
- System Logger
  - cold starting System Logger 15
  - enhancements in z/OS 1.3 and later 13
  - failure independence concepts 67
  - installation and setup tasks 10
  - inventory 11
  - listing the inventory 79
  - MVS DISPLAY commands 44
  - policy 11, 15
    - contents 15
    - differences from other CDSs 15
  - requirement for SMS subsystem 11
  - restart processing 69
  - sequence of log blocks in log stream 19

- serialization 25
- services 16
- System Logger policy
  - AUTODELETE keyword 38, 50
  - DASDONLY keyword 46
  - DESCRIPTION keyword 33, 46
  - DIAG keyword 42, 54
  - DUPLEXMODE keyword 34
  - EHLQ keyword 39, 51
  - HIGHOFFLOAD keyword 40, 52
  - HLQ keyword 39, 51
  - LIKE keyword 41, 53
  - LOGGERDUPLEX keyword 34
  - LOWOFFLOAD keyword 40, 52
  - LS\_DATACLAS keyword 36, 48
  - LS\_MGMTCLAS keyword 37, 49
  - LS\_SIZE keyword 37, 49
  - LS\_STORCLAS keyword 37, 49
  - MAXBUFSIZE keyword 47
  - MODEL keyword 40, 52
  - NAME keyword 33, 46
  - OFFLOADRECALL keyword 42, 54
  - OFFLOADRECALL keyword 264
  - RETPD keyword 38, 50
  - RMNAME keyword 33, 46
  - STG\_DATACLAS keyword 35, 47
  - STG\_DUPLEX keyword 34
  - STG\_MGMTCLAS keyword 35, 47
  - STG\_SIZE keyword 36, 48
  - STG\_STORCLAS keyword 35, 48
  - STRUCTNAME keyword 34
  - updating attributes dynamically 55
- System-Managed Duplexing
  - considerations for System Logger duplexing 68
  - impact of CF volatility changes 74
  - recovery from connectivity failure 73
  - recovery from structure or CF failure 73
  - use with System Logger structures 34, 43

## T

- terminology explained 8
- Transactional VSAM
  - CF versus DASD-only log stream 92
  - coupling facility failure 109
  - criticality of data 87
  - dependency on system logger 105
  - description 84
  - duplexing considerations 88
  - Forward recovery log stream 86
    - LOG(ALL) 86
    - LOG(NONE) 86
    - LOG(UNDO) 86
    - sizing considerations 88
  - Forward recovery log stream definition 91
  - Forward recovery log stream sizing method 89
  - IGDSMS definition 90
  - IGWLOG.SYSLOG 85, 92
  - IGWLOG.SYSLOG CFRM definitions 93
  - IGWSHUNT.SHUNT 92
  - IGWSHUNT.SHUNTLOG 85

- log stream naming convention 84
- log stream permanent full condition 108
- log stream sizing 87
- log stream sizing elements 88
- LOG\_OF\_LOGS log stream definition 91
- log-of-logs log stream 86
- managing Undo log stream 85
- missing data in the log stream 107
- monitoring offload 107
- performance considerations 112
- subsystem failure 112
- system logger failure 110
- temporary log stream full condition 108
- Undo log 85
  - sizing consideration 88
- Undo log description 85
- Undo log sizing method 89
- Transactional VSAM log stream
  - consideration for dedicated structure for the Undo logs 96
  - IGWLOG DASD-only definition 100
  - IGWSHUNT DASD-only definition 100
  - security definitions 104
- Tuning
  - Auto journals 206
  - Forward recovery 206
  - User journals 206

## U

- Unit of Recovery (UR) 84

## W

- WebSphere for z/OS
  - CFRM definitions 249
  - criticality of data 246
  - DASD\_only log stream 250
  - error connecting to the log stream 252
  - log stream definition 247
  - log stream description 246
  - log stream sizing 246
  - logger definition 247
  - logger definitions 249
  - security definitions 251
- WebSphere for z/OS log stream
  - browsing the log stream 251
  - performance considerations 252

## X

- XLGSTRM exit 147
- XRC 307

Archived



## System Programmer's Guide to: z/OS System Logger

Archived









# System Programmer's Guide to: z/OS System Logger



**Redbooks®**

**Managing, optimizing, and sizing the System Logger environment**

**How exploiters can get the most out of System Logger**

**System Logger installation, setup, and usage**

The z/OS System Logger is a function provided by the operating system to exploiters running on z/OS. The number of exploiters of this component is increasing, as is its importance in relation to system performance and availability. This IBM Redbooks document provides system programmers with a solid understanding of the System Logger component and guidance about how it should be set up for optimum performance with each of the exploiters.

System Logger is an MVS component that provides a logging facility for applications running in a single-system or multi-system sysplex. The advantage of using System Logger is that the responsibility for tasks such as saving the log data (with the requested persistence), retrieving the data (potentially from any system in the sysplex), archiving the data, and expiring the data is removed from the creator of the log records. In addition, Logger provides the ability to have a single, merged, log, containing log data from multiple instances of an application within the sysplex.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)**

SG24-6898-01

ISBN 0738489433