

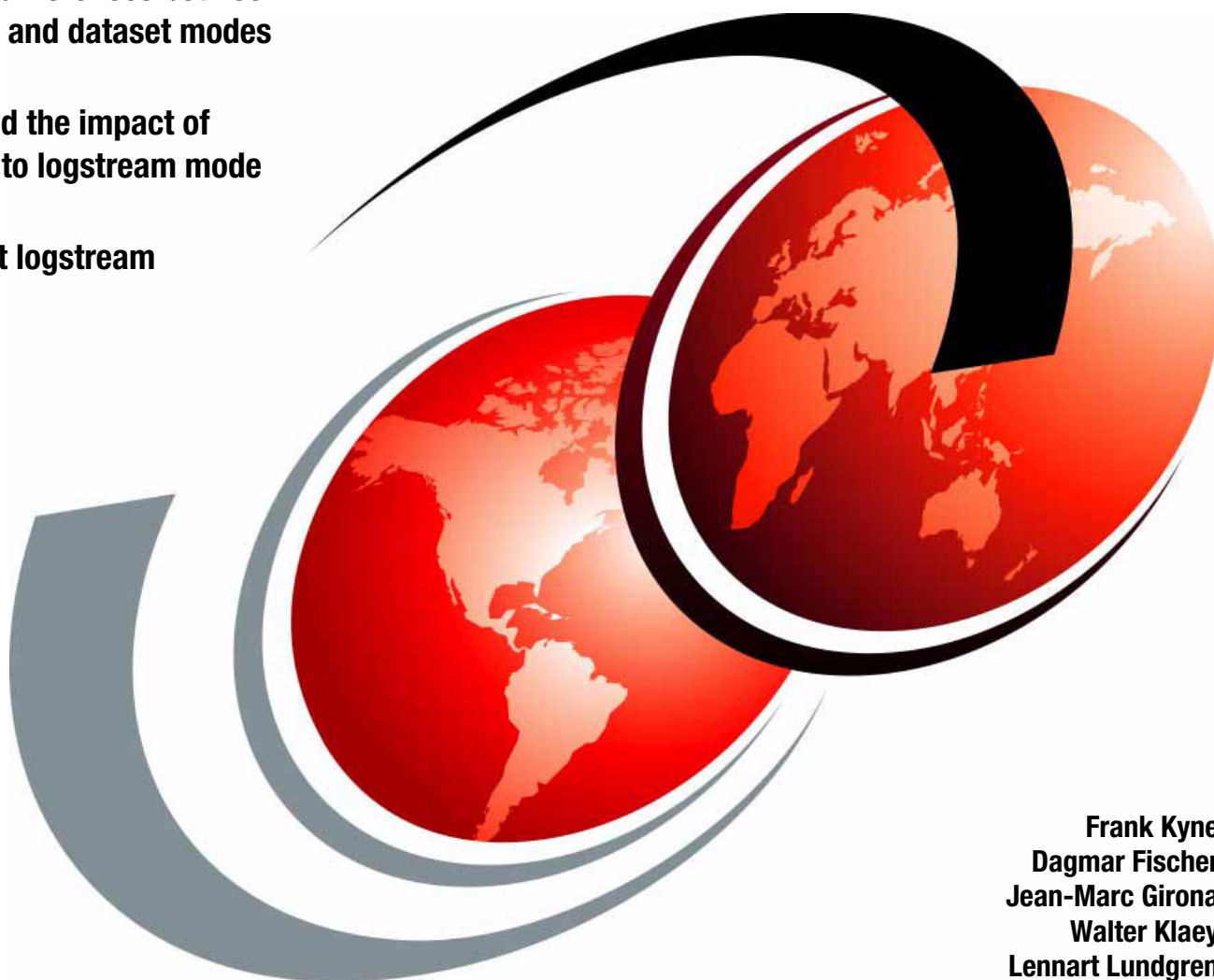
# SMF Logstream Mode

## Optimizing the New Paradigm

Learn the differences between  
logstream and dataset modes

Understand the impact of  
migrating to logstream mode

Implement logstream  
mode



Frank Kyne  
Dagmar Fischer  
Jean-Marc Girona  
Walter Klaey  
Lennart Lundgren

# Redbooks





International Technical Support Organization

**SMF Logstream Mode: Optimizing the New Paradigm**

February 2011

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

**First Edition (February 2011)**

This edition applies to Version 1, Release 12, Modification 0 of z/OS (product number 5694-A01).

**© Copyright International Business Machines Corporation 2011. 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> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team who wrote this book .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	x
Stay connected to IBM Redbooks .....	xi
<b>Chapter 1. Introduction</b> .....	1
1.1 SMF and how it is used .....	2
1.2 How SMF recording originally worked .....	2
1.3 System Logger and how it works .....	6
1.4 SMF support for log streams .....	10
1.5 Migration options .....	16
1.6 Migration considerations .....	17
1.6.1 Benefits of SMF log stream support .....	17
1.6.2 SMF logstream mode considerations .....	22
1.7 Is SMF logstream mode right for you .....	25
<b>Chapter 2. Understanding your environment</b> .....	27
2.1 Why you need to know how your SMF data is used .....	28
2.1.1 Which programs access SMF data .....	28
2.1.2 Which people access SMF data .....	29
2.2 Documenting your current SMF processes .....	30
<b>Chapter 3. Migration approaches</b> .....	37
3.1 Introduction .....	38
3.2 Migration options .....	40
3.2.1 Approach 1: No change .....	41
3.2.2 Approach 2: Replace only the SYS1.MAN data sets with log streams .....	42
3.2.3 Approach 3: Keep certain SMF records in log streams .....	42
3.2.4 Approach 4: Eliminate SMF GDGs .....	44
3.2.5 Migration sequence .....	44
3.3 How many SMF log streams to have .....	45
<b>Chapter 4. Planning</b> .....	49
4.1 Layout of this chapter .....	50
4.2 Preparing for SMF logstream mode .....	50
4.2.1 How many log streams to have .....	50
4.2.2 Which type of log stream to use .....	50
4.2.3 Map out your SMF strategy .....	52
4.2.4 Sizing the SMF log streams .....	57
4.2.5 Preparing SMFPRMxx members .....	64
4.2.6 Review configuration .....	67
4.2.7 Operator education .....	74
4.2.8 Automation considerations .....	75
4.2.9 Install all SMF logstream mode-related service .....	79
4.2.10 System programmer SMF toolkit .....	82

4.2.11 Post-implementation monitoring . . . . .	90
4.3 Planning for approach 2 implementation. . . . .	90
4.3.1 Document existing SMF job network. . . . .	90
4.3.2 Changes to SMF jobs . . . . .	91
4.3.3 Fallback planning . . . . .	93
4.4 Planning for approach 3 implementation. . . . .	93
4.4.1 Archive job changes . . . . .	93
4.4.2 Change log stream definitions . . . . .	94
4.4.3 Provide alternate JCL for processing log streams . . . . .	94
4.4.4 Provide documentation about the location of SMF records by date . . . . .	96
4.4.5 SMF user education . . . . .	97
4.4.6 Handling expired GDGs . . . . .	97
4.4.7 Fallback plan. . . . .	97
<b>Chapter 5. Implementation . . . . .</b>	<b>99</b>
5.1 Preparation . . . . .	100
5.1.1 Adjusting LPAR storage amounts . . . . .	100
5.1.2 Preparing the SMS environment. . . . .	100
5.1.3 Prepare the user catalogs. . . . .	100
5.1.4 Implement automation changes . . . . .	101
5.1.5 Grant RACF access to the SMF log streams . . . . .	101
5.1.6 Adjusting the LOGR couple data set. . . . .	102
5.1.7 Defining the SMF Coupling Facility structures . . . . .	103
5.1.8 Defining the log streams . . . . .	103
5.1.9 Install and activate the SMF IEFU29L exit . . . . .	104
5.1.10 Prepare the SMF IFASMF DL jobs . . . . .	105
5.2 Cutover . . . . .	105
5.2.1 Enable SMF logstream mode . . . . .	105
5.2.2 Verifying successful archive processing . . . . .	106
5.3 Post-migration activities . . . . .	107
5.3.1 Monitoring Logger activity . . . . .	107
5.3.2 SMF buffer use . . . . .	108
5.3.3 Space usage in storage group . . . . .	109
5.3.4 Number and size of offload data sets . . . . .	109
5.3.5 LOGR CDS DSEXTENTS . . . . .	110
5.3.6 Performance of SMF-related CF structures . . . . .	110
5.4 Gotchas. . . . .	110
5.4.1 Difference between ARCHIVE and DUMP . . . . .	111
5.4.2 Issuing T SMF in SDSF. . . . .	112
5.4.3 IFASMF DL support of OPTIONS(ALL) . . . . .	112
5.4.4 Missing offload data sets . . . . .	112
5.4.5 IFASMF DL return codes . . . . .	112
5.4.6 Protecting SMF data in case of System Loggerabend. . . . .	113
5.4.7 Z EOD and I SMF commands. . . . .	113
5.4.8 Importance of sorting SMF records. . . . .	113
<b>Appendix A. Relative capacity measurements. . . . .</b>	<b>115</b>
System Logger performance basics. . . . .	116
Optimizing performance for SMF log streams. . . . .	116
Comparative measurements . . . . .	118
Measurement . . . . .	119
Measurement results . . . . .	120
<b>Appendix B. Important command and message changes . . . . .</b>	<b>123</b>

SMF commands . . . . .	124
SMF messages . . . . .	126
System Logger commands . . . . .	127
Other commands . . . . .	129
Using IPCS to extract SMF records from a dump . . . . .	129
<b>Appendix C. Considerations for systems without APAR OA34589 . . . . .</b>	<b>131</b>
<b>Appendix D. Changing log stream attributes . . . . .</b>	<b>133</b>
<b>Appendix E. Additional material . . . . .</b>	<b>135</b>
Locating the web material . . . . .	135
Using the web material . . . . .	135
Spreadsheets . . . . .	136
Manage_SMF spreadsheet . . . . .	136
Stream_Sizing spreadsheet . . . . .	137
z/OS tools . . . . .	140
RBITSO.CNTL.XMIT data set . . . . .	140
Sample IFASMFDP exits . . . . .	147
How to use the web material . . . . .	150
<b>Related publications . . . . .</b>	<b>151</b>
IBM Redbooks publications . . . . .	151
Other publications . . . . .	151
Online resources . . . . .	151
Help from IBM . . . . .	151
<b>Index . . . . .</b>	<b>153</b>





# 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 on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not 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 and/or changes in the product(s) and/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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

CICS®	Parallel Sysplex®	System z®
DB2®	RACF®	Tivoli®
GDPS®	Redbooks®	WebSphere®
IBM®	Redpaper™	z/OS®
IMS™	Redbooks (logo)  ®	z10™
MVS™	RMF™	
OS/390®	System z10®	

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

# Preface

This IBM® Redbooks® publication positions the use of System Logger log streams as a repository for System Management Facilities (SMF) data against the previous use of Virtual Storage Access Method (VSAM) data sets for SMF data. This book expands on existing material by covering not just the implementation steps, but also by looking at how you use SMF data today, and using that information to help you identify the most appropriate repository for your SMF data.

If it transpires that log streams are appropriate for some or all of your SMF data, this book provides all the guidance that you are likely to require for a successful migration to this new paradigm.

The target audience for this document is system programmers and anyone who uses SMF data.

**Attention:** There have been significant enhancements to SMF since the log stream support was initially delivered in z/OS® 1.9, many of which have been delivered back to z/OS 1.9 or later releases via APARs. This book describes the SMF log stream support as it works if the following APARs are applied:

- ▶ OA27037
- ▶ OA31737
- ▶ OA34589

If the PTFs for these APARs are not currently applied to your system, we suggest that they are applied before proceeding with the migration to SMF logstream mode.

## The team who wrote this book

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 an Executive IT Specialist and Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of Parallel Sysplex® and high availability. Before joining the ITSO 12 years ago, Frank worked in IBM Ireland as an MVS™ Systems Programmer.

**Dagmar Fischer** is a Senior System Programmer at Finanz Informatik Technologie Service in Germany. She has 25 years of experience in the mainframe environment. Her areas of expertise include capacity planning and performance. She regularly presents at GSE Germany and has been a mentor for her younger colleagues.

**Jean-Marc Girona** is a Technical Specialist at Standard Bank South Africa. He has 30 years of experience in the mainframe systems programming field. He worked for Standard Bank for three years. His areas of expertise include Parallel Sysplex support and tuning and systems programming support for many independent software vendor (ISV) software products.

**Walter Klaey** is a Senior IT Specialist in IBM Switzerland. He has 22 years of system programming experience in IMS™, OS/390®, and z/OS on the System z® platform. His areas of expertise include Parallel Sysplex and System Logger.

**Lennart Lundgren** is an IT Specialist in the IBM Software Group, Sweden. He has 30 years of experience in the Systems Management area on mainframe computers. Lennart holds a degree in Computer Sciences from the University of Lund, Sweden. He has worked at IBM for more than 20 years and his areas of expertise include performance and capacity management, accounting and charge back, Tivoli® Decision Support use and customization, and tools development.

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

Bob Haimowitz  
International Technical Support Organization, Poughkeepsie Center

Mario Bezzi  
IBM Italy

Concha Tourné Izquierdo  
IBM Spain

Marianne Hammer  
Andrea Harris  
Steven B Jones  
Bonnie Ordonez  
Andrew Sica  
Anthony Sofia  
Frank Yaeger  
Doug Zobre  
IBM USA

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

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

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

- Send your comments in an email to:

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

- Mail your comments to:

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

## Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>





# Introduction

This chapter reviews System Management Facilities (SMF) processing, starting with how it works when writing to Virtual Storage Access Method (VSAM) data sets (now referred to as *dataset mode*). It then provides an overview of the ability to write SMF records to a log stream (known as *logstream mode*) and discusses the background behind why this function was introduced. It also summarizes the benefits and restrictions of each approach and provides information to help you determine which methodology is best suited to your environment. By the end of this chapter, you will know whether you will continue using SMF in dataset mode or whether the use of log streams is attractive enough to warrant further investigation.

**Attention:** There have been significant enhancements to SMF since the log stream support was initially delivered in z/OS 1.9, many of which have been delivered back to z/OS 1.9 or later releases via APARs. This book describes the SMF log stream support as it works when the following APARs are applied:

- ▶ OA27037
- ▶ OA31737
- ▶ OA34589

If the PTFs for these APARs are not currently applied to your system, we suggest that they be applied before proceeding with the migration to SMF logstream mode.

## 1.1 SMF and how it is used

The z/OS MVS SMF gathers and records information about events and resource usage in your system. This information is saved in timestamped records and grouped by subsystem, providing you with unparalleled insight into the activities of your z/OS system. SMF information is commonly used to generate reports for performance management, storage management, security violations, database performance, resource utilization, system event reports, and much more. There are many different reporting packages available from IBM and other vendors.

## 1.2 How SMF recording originally worked

To fully understand the changes brought about by the System Logger-related enhancements in SMF and how they can impact you, it is important to understand how SMF works when writing to its SYS1.MAN data sets (dataset mode) and how the SMF data is subsequently managed and used. Many readers will already be familiar with this, but it is still valuable to review it briefly.

As shown in Figure 1-1, programs write records to SMF using SMFWTM or SMFEWTM macros. The records are grouped together in a buffer in the SMF address space, and then written to a Control Interval in the SMF VSAM data set (SYS1.MAN).

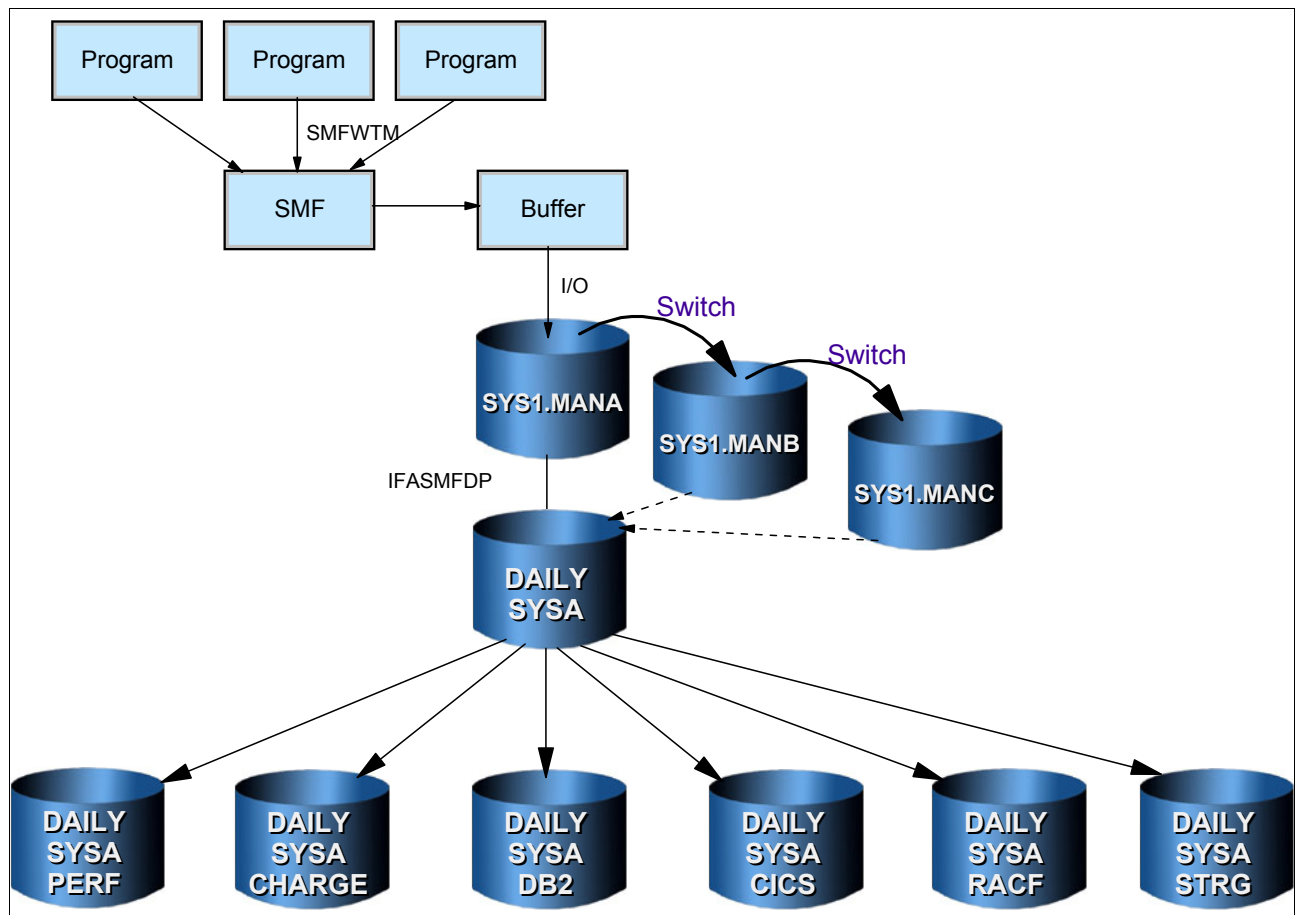


Figure 1-1 Overview of SMF data flow



As records are received by SMF, they are held until either the MAXDORM time is reached or enough data is received to fill a Control Interval in the SYS1.MAN data set, whichever happens first. The records are placed in the block in the order in which they arrive.

For SMF record types 1 to 127, the SMFxxDTE (date), SMFxxTME (time), and SMFxxSID (system ID) fields are filled in by SMF when the records are passed to it. For user SMF records (types 128 to 255), those fields are filled in by the program that creates the record. These are the fields that are normally used if you need to sort SMF records into chronological order.

If the record being sent to SMF would cause the block to contain more than 32 KB, part of the record will be written to the current block, and the remainder will be placed in the next block. That is, when writing to a SYS1.MAN data set, SMF records can span multiple blocks.

As you can imagine, the rate at which SMF records are created varies. Certain records get created whenever an event happens, for example, when a data set is opened. Other records get created on a timed basis. For example, RMF™ records might be created on the hour and then every 15 minutes, so it is reasonable to expect a large volume of records to be created at certain times, such as on each hour.

The volume of SMF data created depends on many things, including the capacity of the system, the mix of software products, the SMF interval that you specified, and which SMF record types and subtypes you have enabled. Because of the wealth of information contained in the SMF records, it is understandable that the system programmers responsible for a certain aspect of z/OS would want to enable and collect the records related to the products for which they are responsible. There are many good reasons why you would want to collect large volumes of SMF data.

Eventually, the SYS1.MAN data set fills up and SMF automatically switches to the next empty SYS1.MAN data set. When the switch occurs, exit IEFU29 is driven and a message is sent to the console, informing you that the data set has filled and switched. Most sites use one of these events to kick off a process to extract all data from the full SYS1.MAN data set and then clear that data set, making it available for use again.

It is possible that SMF records are being created faster than they can be moved from the SMF address space to the SMF data set. This is where the SMF buffers come into play. You can have between 128 MB and 1 GB of SMF buffers when using SMF in dataset mode (depending on the setting of the BUFSIZE keyword in SMFPRMxx), so it is possible for SMF to handle large disparities between the rate at which SMF data is created and the rate at which it can be moved to DASD. However, if a program attempts to pass a record to SMF and all the buffers are already full, one of two things will happen (depending on the setting of the NOBUFFS parameter in the SMFPRMxx member):

- ▶ A message will be issued to inform you that all SMF buffers are full, and subsequent SMF records will be discarded until space becomes available in the buffer<sup>1</sup>. This is the action if you specify NOBUFFS(MSG).
- ▶ If you specify NOBUFFS(HALT), the system enters a restartable wait state. If you decide to restart the system, when it comes back, the buffers will still be full, and new records sent to SMF will be discarded until space becomes available in the buffer.

If your enterprise has determined that no SMF data loss is acceptable, you would have to take a stand alone dump rather than restarting the system, and subsequently retrieve the SMF records that were in the SMF buffer from the dump.

---

<sup>1</sup> SMF Record Type 7 contains information about the number of discarded records.

Note that when SMF is in dataset mode, there is no granularity related to controlling these actions. Only one action can be specified for a system, and all records will be treated in a similar manner.

Because it is impossible to know in advance which records will arrive after the buffers fill (and therefore will be discarded), if there are certain SMF record types that you *absolutely must not* lose, the only way to ensure that those record types do not get discarded (other than specifying NOBUFFS(HALT) and taking a standalone dump) is to tune down the number of record types that you collect in an attempt to ensure that the SMF buffers never fill. This means turning off the collection of certain SMF data that you otherwise would like to collect.

As z/OS systems get larger and the number of transactions being processed continues to increase, more and more customers face this situation. Also, processor speeds and capacities continue to grow at a faster rate than DASD speeds, making it likely that the creation rate of SMF data will continue to outstrip the ability of the SYS1.MAN data sets to keep up. This was one of the primary drivers for the enhancement to let SMF write its records to log streams rather than the SYS1.MAN data sets.

Note that the flood automation support introduced with z/OS 1.12 is no guarantee of not losing data. Specifying FLOOD(ON) in SMFPRMxx allows you to define what you regard as an unacceptably high arrival rate of records (that is, a flooding condition). When a flooding condition is detected, you have the option of displaying a message and optionally dropping records. Flood automation is like an early warning system and gives you more time to take appropriate action. However, the utilization of the SMF buffers is *not* considered by the flood automation function. Hence, it should not be viewed as a solution to the problem of SMF discarding records when its buffers fill.

There are other characteristics of SMF use of SYS1.MAN data sets that are perhaps not what you would ideally like:

- The SYS1.MAN data sets contain all SMF record types for a given system. However, most enterprises want a sysplex-wide view of a given subset of SMF record types (Figure 1-2). To address the difference between how SMF data is collected and how it is used, customers typically gather a given SMF record type or types for the entire sysplex into a set of GDGs and process the data from there. This means that a certain amount of postprocessing is required to group the SMF records in the way that you want to use them.

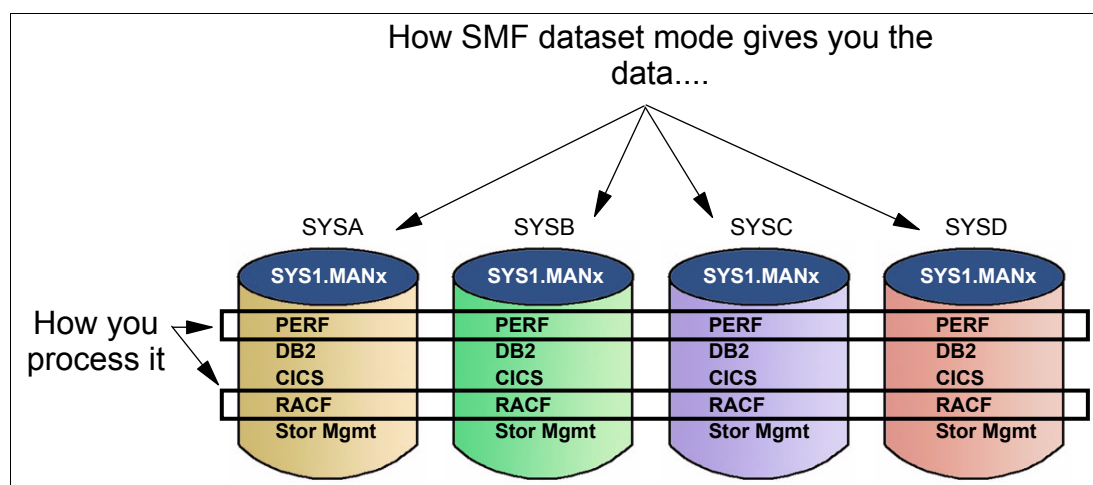


Figure 1-2 How SMF records are stored in SMF dataset mode

- SMF does not support extended format data sets. This means that SMF cannot exploit extended format attributes such as greater than 4GB support, or data set striping.

- If you have some SMF record types that are critical to your organization, you cannot create a second copy of that data until the records are extracted from the SYS1.MAN data sets.

SMF data is usually dumped from the SYS1.MAN data set automatically when there is a data set switch, at which point a new generation of a GDG is created and the SYS1.MAN data set is cleared. Or you may have procedures to create a new data set with a data set name containing the timestamp and the system name for easy identification. In any case, you most likely create a new set of sequential data sets containing SMF records for a given system for each SMF data set switch.

Typically, the SMF records are processed by multiple applications. Those applications probably only require a subset of the records, however they may require the input data set to contain SMF records from several systems (the scope is usually all the systems in the sysplex, because those systems typically have a logical relationship). To create the input for these applications, you have to extract the selected record types from the GDG for each system and then merge them into a new sysplex-wide file for processing. Figure 1-3 shows an example of the flow of processing for SMF data, including the splitting of SMF data across multiple GDGs and subsequent merging into a sysplex-wide file.

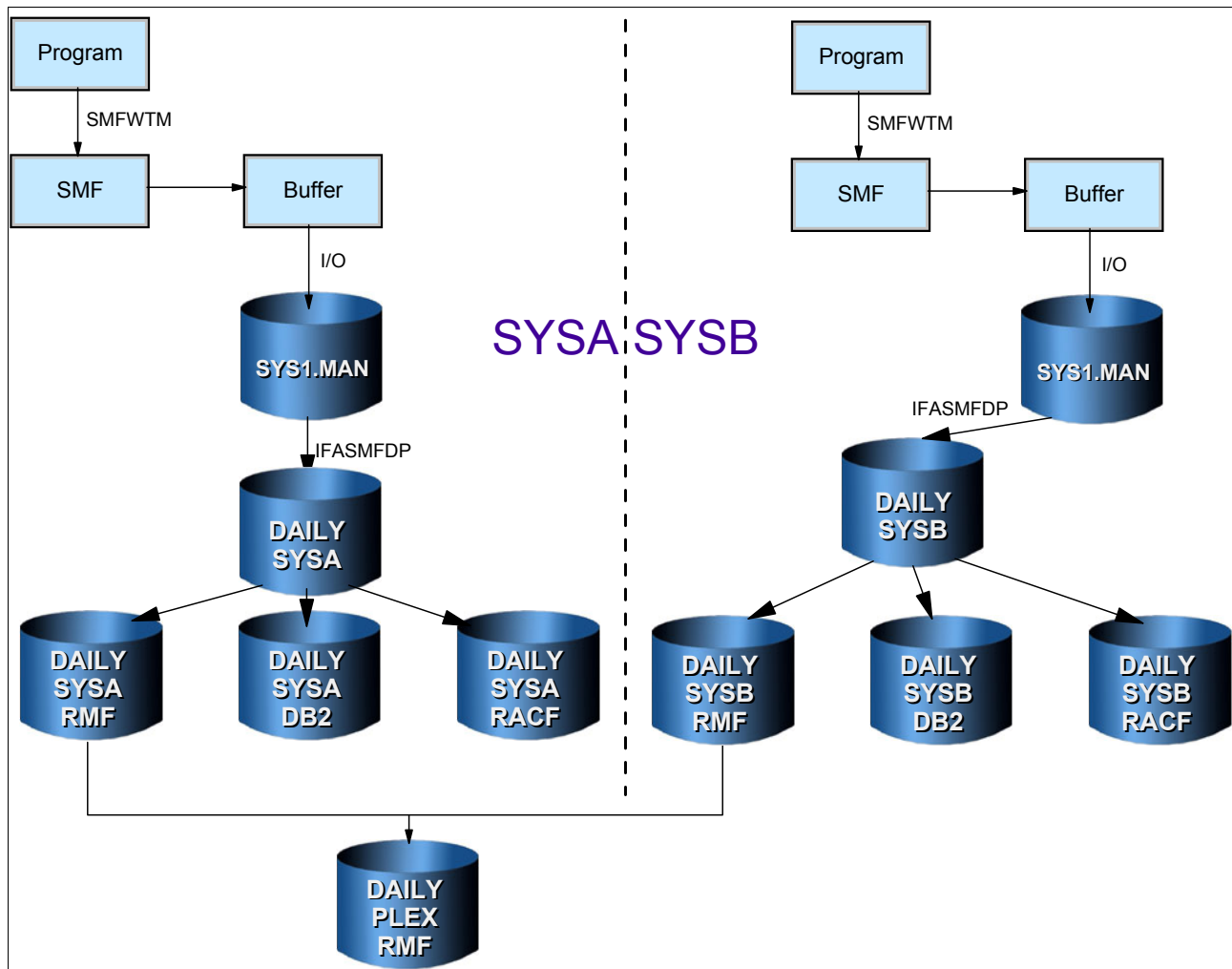


Figure 1-3 Flow of SMF data

Because of the large number of generations, different data combinations, retention periods, and legal requirements to keep logs, managing SMF data can become quite complex, and the processes and procedures that you have built up over the years reflect that complexity. Helping you understand and manage that complexity is one of the objectives of this book.

## 1.3 System Logger and how it works

System Logger provides a generalized logging facility for subsystems and applications running in a single-system or multi-system sysplex. System Logger provides the ability to easily store, retrieve, and delete log records. Advantages of using System Logger, rather than creating your own logging function, are:

- ▶ The ability to easily have a single, sysplex-wide repository of log records.
- ▶ Easier recovery. It is Logger's responsibility to recover from error situations.
- ▶ Improved resiliency for log records based on availability requirements that you define to System Logger, and Logger's understanding of the sysplex configuration.
- ▶ Providing the optimum performance within the constraints of delivering the levels of availability that you require.
- ▶ Easier management of log data.

System Logger exploiters use a system service to pass blocks of data to Logger. These blocks of data can contain one log record or many log records, as decided by the exploiter. The blocks of data, *log blocks* in System Logger terminology, are stored by Logger in a log stream. Logically, and from the perspective of the exploiter, a log stream is simply a long stream of log blocks that can be accessed via System Logger calls. Note that System Logger works at the log block level—it has no knowledge or understanding of the data inside a log block—so requests to System Logger to write, retrieve, or delete are at the log block level, not at the individual log record level.

Each log block has a unique identifier known as a log block ID, and a timestamp that represents the time that the block was written to the log stream. Logger guarantees that the timestamp of each block in the log stream will be greater than or equal to the timestamp of the preceding log block in the log stream. This means that as you read forward in a log stream, even one containing log blocks from multiple systems, the timestamp for each sequential log block will never move backwards.

Physically, the most recent part of the log stream exist in at least two locations (these, collectively, are referred to as *interim storage*). The two locations are a combination of:

- ▶ A structure in a CF
  - If a log stream will be used by exploiters in more than one system, it *must* be defined as a CF log stream, meaning that one of the copies of the most recent log records will reside in a CF structure.
- ▶ A System Logger data space
- ▶ A System Logger staging data set

Which two of these locations are used depends on how you define the log stream and the sysplex configuration. Figure 1-4 shows the relationship between the physical and logical views of a log stream.

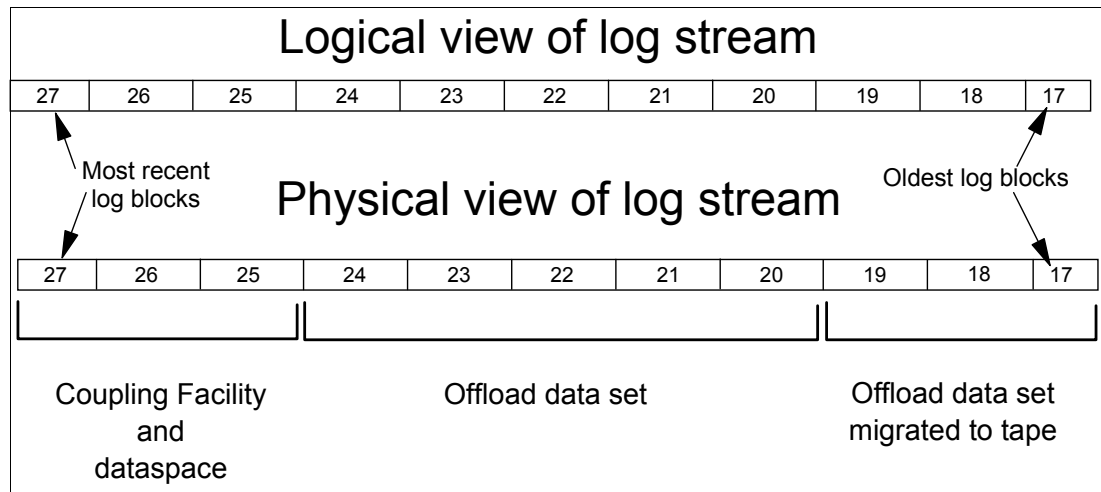


Figure 1-4 Logical and physical views of log stream

As more records are written to the log stream, older records that have not yet been deleted will be moved from interim storage to offload data sets. As the data gets still older, it might eventually be migrated to tertiary storage by a product like DFSMSHsm. However, regardless of where a given log block physically exists, it is always accessible by a call to System Logger, and the exploiter is completely unaware of its physical location.

Eventually, log blocks will be deleted, either by the exploiter or by System Logger (based on the retention period that you specify when you define the log stream). Note that it is not possible to have gaps in the log stream. For example, looking at the log stream in Figure 1-4, you cannot delete log block 19 but keep log block 17. If you request System Logger to delete log block 19, it deletes log block 19 *and* all older log blocks (18 and 17, in this example).

The deletion of log blocks is a two-step process. When the exploiter deletes a log block, the block is *logically* deleted from the log stream. However the *physical* deletion of the log block only happens later, when the log stream reaches its HIGHOFFLOAD threshold, at which point an offload is initiated and the physical deletion of logically deleted log blocks occurs<sup>2</sup>.

<sup>2</sup> If you specify a non-zero RETPD for the log stream, log blocks are not actually be deleted from the log stream until the retention period expires.

Figure 1-5 brings all this together and summarizes the interaction between a number of exploiters and System Logger.

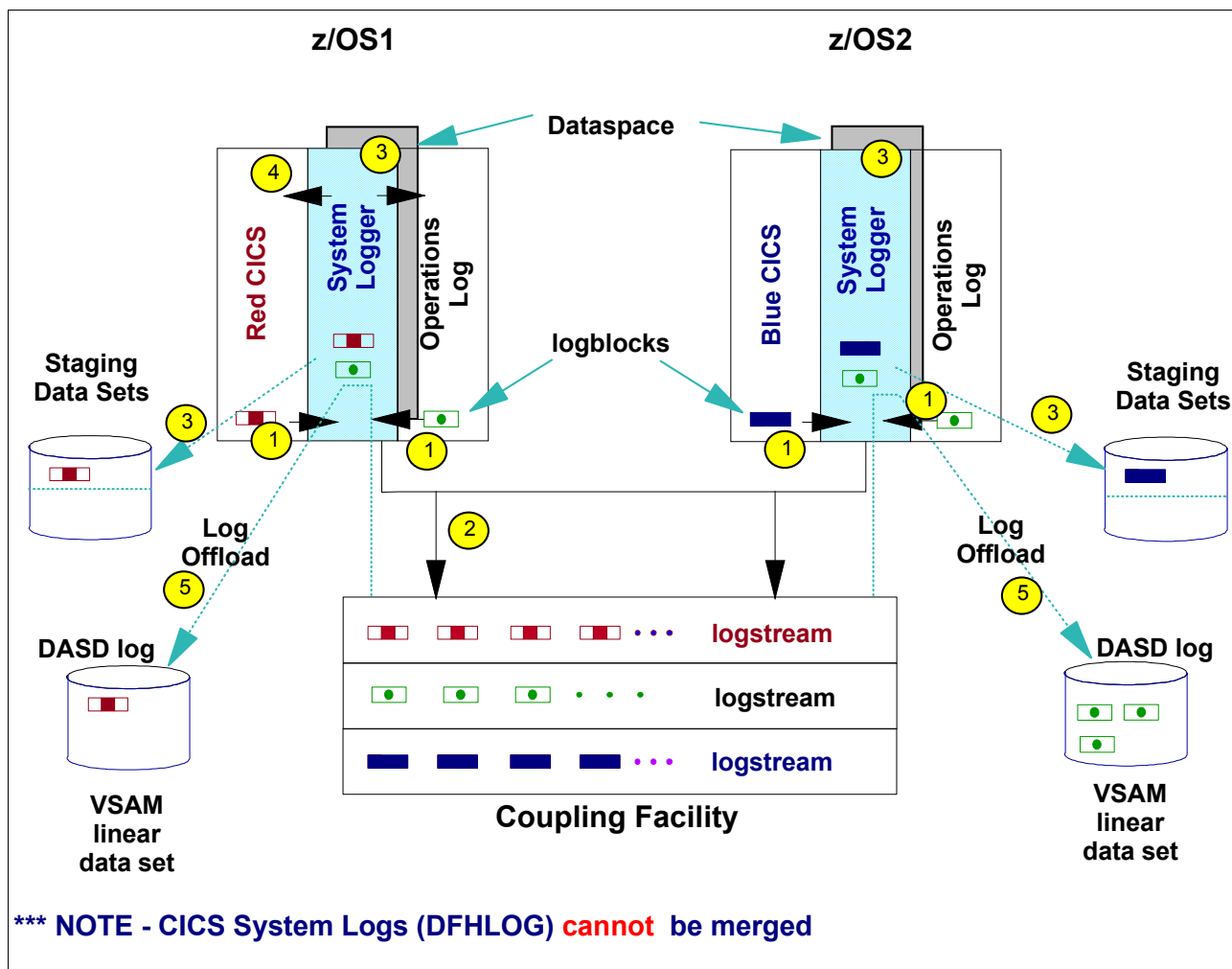


Figure 1-5 System Logger overview

The activities shown in Example 1-1 on page 10 are as follows:

- Both the red and blue CICS® regions use an IXCWRITE call to pass a block containing CICS DFHLOG log records to System Logger. Because CICS uses dedicated log streams for DFHLOG log records, the red CICS region has one log stream, and the blue CICS region has another log stream.  
You will see that operations log (OPERLOG) is also writing log records to the System Logger address space. One of the objectives of OPERLOG is to give operators a sysplex-wide view of the MVS syslogs, so you can see that there is only one OPERLOG log stream in the CF, containing the syslog records from both systems.
- Because the installation has defined the DFHLOG log streams and the OPERLOG log stream as CF log streams, the log blocks that have just been passed to System Logger are written to the log streams in the CF.

3. The installation has specified that the DFHLOG log streams are critical, so the second copy of the log blocks will be written to staging data sets (one for the red CICS and one for the blue CICS).

On the other hand, because the data in OPERLOG is only a copy of syslog, the installation has specified that the OPERLOG log stream is less critical. Therefore, the second copy of the OPERLOG log blocks is kept in a System Logger data space in each system.

4. In this example, the red CICS region needs to retrieve a log record, perhaps to back out a transaction, so an IXGBRWSE call is issued to Logger to get that log block back.
5. To free up room in the CF structure, log blocks from the red CICS and from the OPERLOG log streams are moved to offload data sets during a System Logger process known as *offload*.

There are basically two ways in which a System Logger exploiter can use a log stream:

- ▶ It writes log records and either never deletes them or it deletes them in large blocks a relatively long time after they are written. These are called funnel-type log streams.
- ▶ It writes log records, and then deletes them soon afterwards. These are called active-type log streams.

Examples of funnel-type log streams are:

- ▶ Logrec
- ▶ Operlog
- ▶ z/OS Health Checker
- ▶ IMS Shared Message Queue
- ▶ SMF

Examples of active-type log streams are:

- ▶ APPC
- ▶ CICS DFHLOG and DFHSHUNT log streams
- ▶ Certain RRS log streams

In the example in Figure 1-5 on page 8, there is no movement of log blocks from the blue CICS to an offload data set. The design of CICS is that it writes a log record at the beginning of a transaction, and then deletes it when the transaction ends. Because there is no need to save deleted log blocks, the space freed up from deleted log blocks can be used by new log blocks. One of the design objectives of a CICS log stream is that the CF structure should be large enough to hold log blocks until they are deleted, rather than moving them to an offload data set and then deleting them a short time later.

SMF, on the other hand, never deletes log blocks from its log streams, so the objective with an SMF log stream is generally to move the log blocks from interim storage to offload data sets as quickly and efficiently as possible.

The definitions of the log streams that you will use, along with information about the location of all log blocks, are kept in the Logger CDS.

For more information about System Logger, see the IBM Redbooks publication *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

## 1.4 SMF support for log streams

z/OS 1.9 introduced the ability to write SMF records to one or more log streams. This is known as logstream mode. The primary driver for this new capability was that the SYS1.MAN

data sets are becoming a bottleneck for certain installations and a better-performing alternative needed to be found.

Apart from the fact that writing to a log stream is generally faster than writing to a SYS1.MAN VSAM data set, logstream mode adds the ability to have *multiple* log streams for SMF records, as opposed to the single VSAM data set that is supported in dataset mode. This provides a level of scalability that is not possible when writing to a single VSAM data set.

If you are using SMF in logstream mode, you can specify multiple log stream names, and control which record types will be sent to each log stream. This is specified on the LSNAME parameter in the SMFPRMxx member (Example 1-1). In fact, Example 1-1 shows how the SMFPRMxx member can contain both the log stream names and the SYS1.MAN names. This is intended to allow you to easily move back and forth between dataset mode and logstream mode should you need to. Note that it is *not* possible to have a single system writing to both SYS1.MAN data sets *and* log streams at the same time.

*Example 1-1 Defining log streams in SMFPRMxx member*

---

```

ACTIVE                /* ACTIVE SMF RECORDING                */
RECORDING(LOGSTREAM)  /* SMF LOGR RECORDING ACTIVE          */
DSNAME(SYS1.&SYSNAME..MANC, /* SMF DATA SET NAMES              */
        SYS1.&SYSNAME..MAND) /* SMF DATA SET NAMES              */
DEFAULTLSNAME(IFASMF.&SYSPLEX..DFLT)
LSNAME(IFASMF.&SYSPLEX..DB2,TYPE(100:102))
LSNAME(IFASMF.&SYSPLEX..CICS,TYPE(110))
LSNAME(IFASMF.&SYSPLEX..MQ,TYPE(115,116))
LSNAME(IFASMF.&SYSPLEX..TCP,TYPE(118,119))
LSNAME(IFASMF.&SYSPLEX..SECU,TYPE(80:83))
LSNAME(IFASMF.&SYSPLEX..RMF,TYPE(70:79))
LSNAME(IFASMF.&SYSPLEX..CHRGBACK,TYPE(30))

```

---



Another fundamental change introduced by the ability to write SMF records to log streams is that those repositories can contain data from more than one member of the sysplex (whereas the SYS1.MAN data sets can only be in use by one system at a time). It is common to see SMF data being read and moved a number of times before it reaches its final home in the same data set as similar records from every other member of the sysplex. If you set up SMF so that multiple systems write a particular set of SMF record types to the same log stream, then the records are *already* merged, potentially resulting in a time and resource saving. Figure 1-6 illustrates this concept.

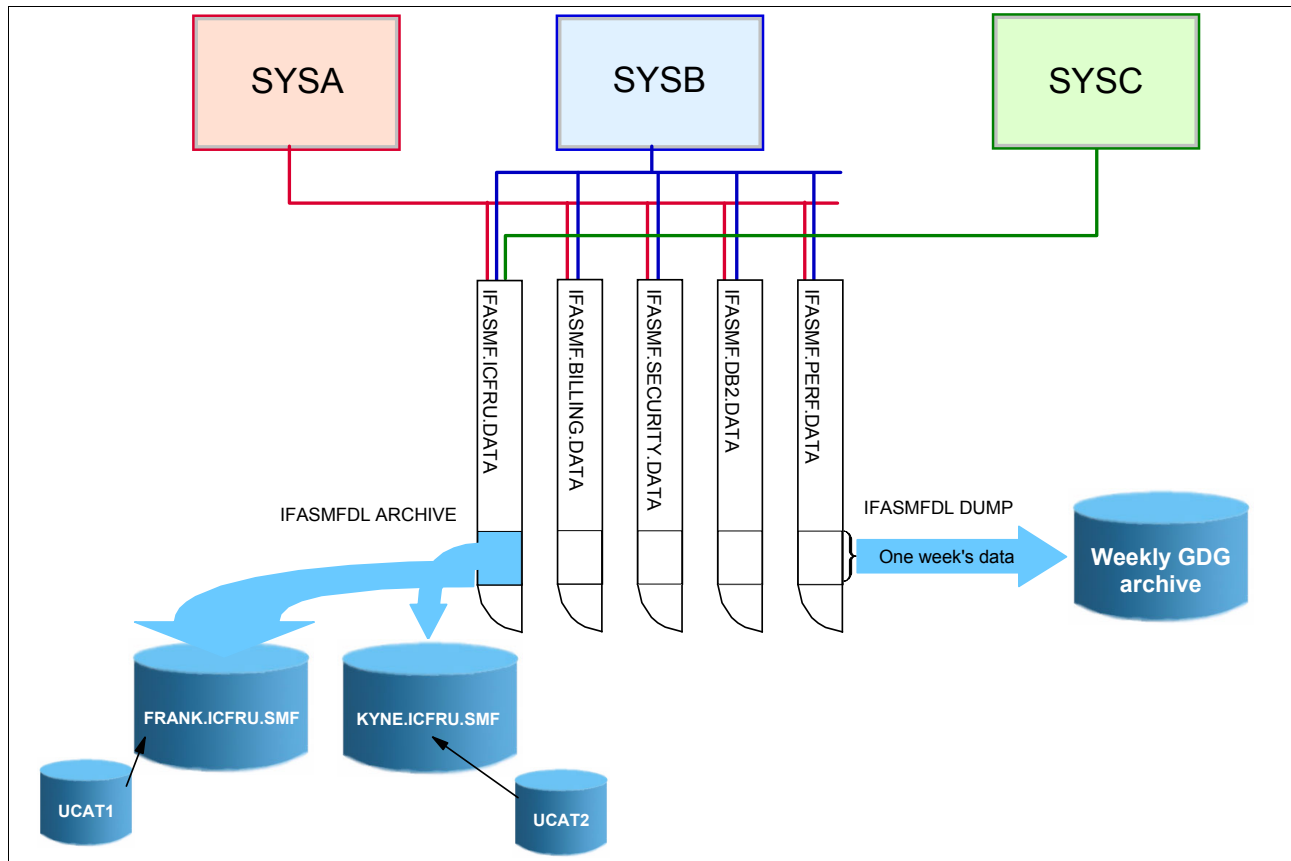


Figure 1-6 Sysplex-wide log streams

Another change introduced as part of the log stream support is that the SMF buffers now reside in one or more data spaces (one per log stream), meaning that the maximum buffer space is now 2 GB *per log stream*, rather than the maximum of 1 GB that is available when writing SMF records to the SYS1.MAN data sets. So, not only do you benefit from shorter response times compared to the VSAM data sets, you also get more and larger buffers, allowing even larger discrepancies between the rate at which records can be created and how quickly they can be externalized.

Related to the support for multiple output destinations is an increase in the number of tasks in SMF that handle the writing of SMF records. When running in dataset mode, SMF has one task to receive SMF records from applications (which happens at memory speeds), and one task to write those records to the SYS1.MAN data sets (which happens at DASD speeds). When running in logstream mode, SMF still has one task to receive SMF records. However, it now has one task *per log stream* to handle the writing of the log blocks. Given the disparity between memory speeds and I/O speeds, it makes sense that there would be more tasks on the writing side of SMF than on the receiving side.

SMF can use either CF or DASDONLY log streams. If you want to consolidate SMF records from multiple systems into one log stream, then you *must* use a CF log stream. Otherwise, you have a choice between DASDONLY and CF log streams. Which one is the most appropriate for a given log stream is discussed in more detail in 4.2.2, “Which type of log stream to use” on page 50.

Whereas the contents of the SMF records are identical regardless of whether the records are written to a VSAM data set or a log stream, the grouping of records into blocks is a little different. When SMF is writing to a VSAM data set, it prefers to always write full control intervals in an attempt to make the I/Os as efficient as possible. If a buffer already contains 31 KB of data and a new 1500 byte record arrives, SMF puts the first 1024 bytes into the block, then starts a new block with the remaining 476 bytes.

On the other hand, when the SMF data is written to a log stream, SMF does *not* span a record over multiple blocks. This means that each block passed to System Logger probably has a block size that is a little different from the previous block. In addition, Logger supports log block sizes up to nearly 64 KB—twice as large as the blocks that are written to the SYS1.MAN data sets.

That brings us to the process of emptying the SYS1.MAN data sets into sequential data sets. IBM provides a program called IFASMFDL that will read a data set (either a SYS1.MAN or a sequential data set) and extract SMF records into one or more sequential data sets. IFASMFDL cannot read a log stream, so a new program called IFASMFDL was provided that fulfills a logically equivalent role to IFASMFDL.

However, there are a number of differences between how IFASMFDL and IFASMFDL are used to manage your SMF data. IFASMFDL provides three options for handling the input VSAM files:

<b>DUMP</b>	Copy data from the data set but do not delete the records after they are read.
<b>CLEAR</b>	Delete all records from the data set and mark the data set as being available for reuse.
<b>ALL</b>	Dump all the records from the data set, then delete all the records.

It is important to remember that the input data set is no longer being written to when you run the IFASMFDL program, whereas the log stream that is input to the IFASMFDL program is constantly being written to.

When SMF support for log streams was originally introduced in z/OS 1.9, the only option for IFASMFDL was DUMP<sup>3</sup>, which provides approximately the same function as the DUMP option in IFASMFDL. However, there was no equivalent to the CLEAR or ALL options. This meant that the processes that used IFASMFDL previously could not in all cases just be converted over to use IFASMFDL instead.

Consider how the SYS1.MAN data sets are typically processed. In most cases, all records are extracted to one or more sequential data sets, and the SYS1.MAN data set is then cleared (this is achieved using the ALL option). Because all records are being extracted, you do not need to specify a start or end date or time, and if you accidentally run another IFASMFDL against the same data set, the data set will be empty, so you should not end up with duplicate records in the output file.

Then consider the log streams. Because there was originally no way to delete log blocks from the log stream as part of the extraction process, every time that you ran an extract (DUMP),

---

<sup>3</sup> The ALL keyword is supported with IFASMFDL, but the function is identical to the DUMP option.

you had to specify a start and end time and date. Otherwise, you would get duplicate records in the output data sets.

To address this shortcoming, z/OS 1.11 introduced two enhancements to IFASMFDP that give you alternatives for how you manage the log stream (these enhancements were also rolled back to z/OS 1.9 and 1.10 with APAR OA27037).

- One enhancement is the addition of two new keywords for the OPTIONS parameter:
  - DELETE
  - ARCHIVE

ARCHIVE is similar (but not quite identical) to the ALL option for IFASMFDP. Specifying ARCHIVE results in all the log blocks from the beginning of the log stream *regardless of any start date or time you specify*, up to the end date and time that you specify, being moved to the sequential data sets that you specify on the OUTDD statements, and then being deleted from the log stream. However, unlike the ALL or CLEAR options, ARCHIVE does not delete *all* log blocks from the log stream, it only deletes the log blocks that it has successfully copied to an OUTDD data set.

- The other enhancement to IFASMFDP, intended to let you process the SMF log streams in a manner that is more consistent with how you process the SYS1.MAN data sets today, is the addition of a new RELATIVEDATE keyword. If you decide to keep certain SMF record types in log streams until they expire, you would get duplicate records if you were to just run an IFASMFDP DUMP job with no start or end date or time against the log stream multiple times. One way around this is to change the START and END dates and times every time that you run a DUMP job. However, this requires additional programming if you want to be able to submit the DUMP jobs in an automated manner. Instead, you can use the new RELATIVEDATE support, whereby the SMF records that will be extracted from the log stream will be for a time period relative to the date that the job is run. So, for example, if you want to run a report against yesterday's data, you can specify RELATIVEDATE(BYDAY,1,1), indicating that you want to extract one day's worth of records, starting at 00:00:00 yesterday. You can also specify RELATIVEDATE in terms of weeks or months.

**Tip:** The use of RELATIVEDATE(BYDAY,0,1) is a special case.

For an ARCHIVE or DELETE operation, the BYDAY,0,1 value indicates that the start time of the operation will be the beginning of the log stream, and the END time for the operation should be set implicitly to the time that the program started executing. If you do not use RELATIVEDATE, the end time of the operation will be set to December 31, 2099, unless you explicitly specify an end date and time.

For a DUMP operation, specifying BYDAY,0,1 indicates that the start time of the operation will be midnight of the current day (00:00:00), and the END time for the operation should be set to December 31, 2099 or the end of the log stream, whichever comes first.

A difference between logstream and dataset mode is how the use of log blocks can subtly change how extracting and deleting records works. When SMF passes a block of data to System Logger, that becomes a log block. A log block has a blockid and a timestamp associated with it, with the timestamp being the time at which the block was written to the log stream.

Now consider the situation in which many systems are writing to a single log stream in a CF, and this log stream happens to usually (but not always) have a low level of activity against it, meaning that each log block can potentially contain SMF records spanning a long time. Figure 1-7 shows a number of log blocks in such a log stream. You will see that the log block timestamps are ever-increasing, even though the timestamps on the SMF records within successive log blocks might be moving forward and backward in time.

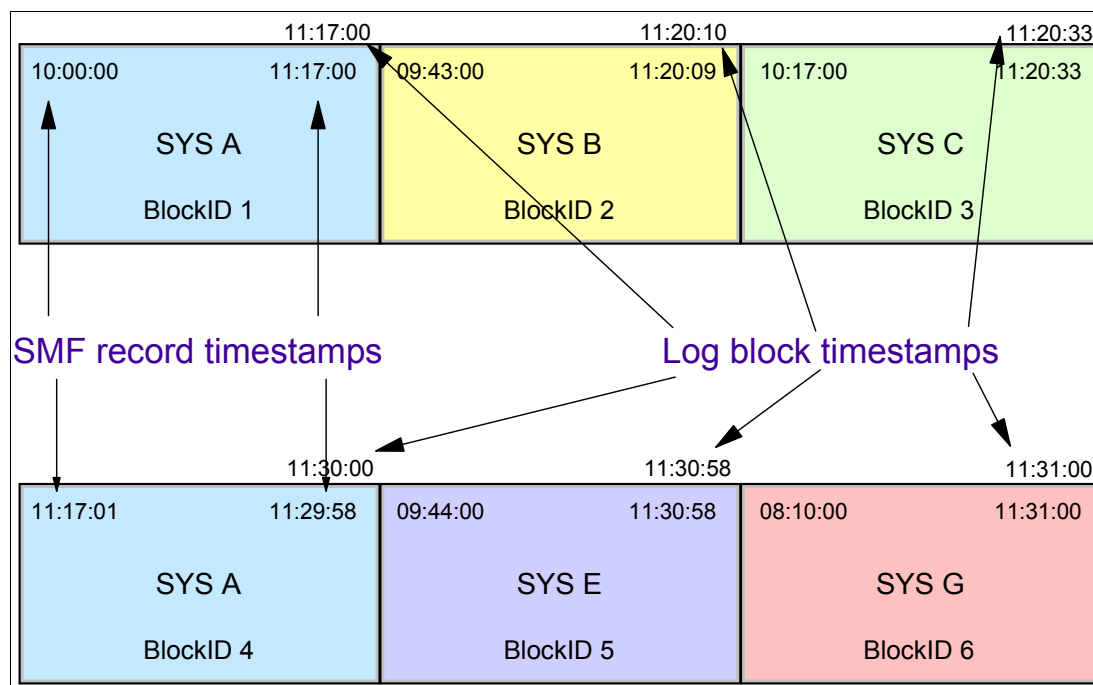


Figure 1-7 Log blocks in a multi-system CF log stream

When log blocks are retrieved from a log stream using IFASMFDDL DUMP, the timestamp of the SMF records within the log blocks are used. Using the log blocks in Figure 1-7 as an example, what would happen if you are looking for SMF records between 00:00:00 and 11:00:00? BlockID 1 has records from system SYSA up to 11:17:00. Does that mean that it is not necessary to read any more log blocks? If we stop at this point, you miss qualifying records from systems SYSB, SYSC, SYSE, and SYSG. Therefore, IFASMFDDL needs to continue reading forward through the log stream, looking for other log blocks that contain SMF records that meet your selection criteria. But how far forward should it go?

Originally, IFASMFDDL read all the way forward, right up to the most recent log blocks. If the log stream only contains data for a small amount of time, this is probably not a big deal and this behavior might not even be noticed. But what happens if the log stream contains two years' worth of data, and you are trying to extract data for just one day, but from six months ago? In this case, the impact is likely to be more noticeable, especially if you plan to migrate old offload data sets to tape because every offload data set would need to be recalled from tape, even though you do not actually want the data in those offload data sets.

To address this situation, a new capability known as SMARTENDPOINT was added to the DUMP function in IFASMFDDL by APAR OA31737 (available back to z/OS 1.9). SMARTENDPOINT indicates that processing of input records in the log stream should cease once it has been determined that records for all known SYSIDS contain a date and time that is some time past the IFASMFDDL specified end date and time. By default, that time is two hours, but APAR OA34374 provides the ability to override that value. Note that SMARTENDPOINT is not the default. If you want to use this function, you *must* explicitly specify it in your IFASMFDDL SYSIN statements<sup>4</sup>.

A related consideration is about the granularity at which objects can be deleted from Logger. Because Logger has no understanding of the contents of a log block, it is not possible for it to delete half the contents of a log block. Only full log blocks can be deleted. This means that the behavior of the DUMP option in IFASMFDL is different from that of ARCHIVE and DELETE. Consider the log blocks shown in Figure 1-8 and that each log block has SMF records from the following times:

<b>Block 1</b>	2010.200 2200 to 2010.200 2259
<b>Block 2</b>	2010.200 2300 to 2010.201 0059
<b>Block 3</b>	2010.201 0100 to 2010.201 0259
<b>Block 4</b>	2010.201 0300 to 2010.201 0459

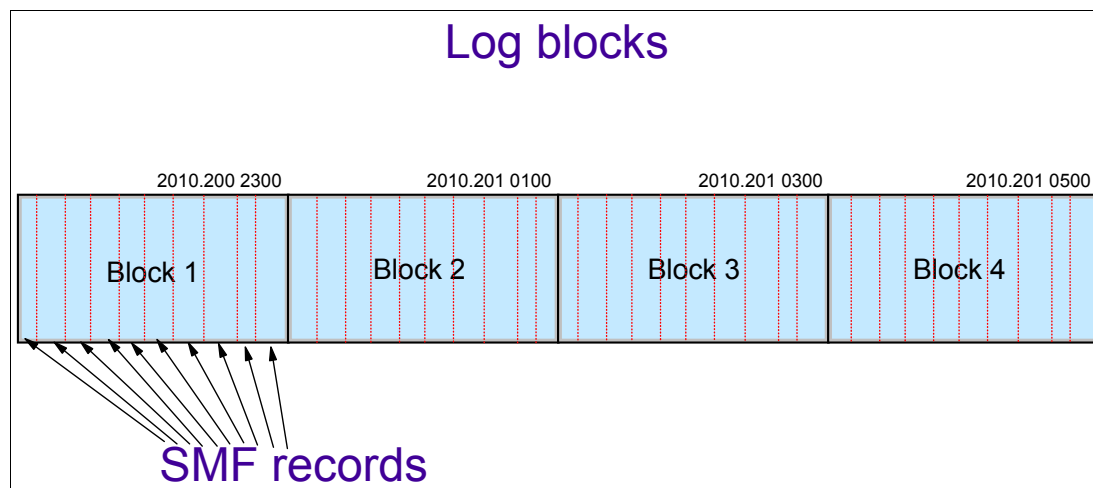


Figure 1-8 Relationship of SMF records to log blocks

Because the DUMP function returns SMF records at the record level, if you run a DUMP job asking for all records up to 2010.200 2330, you will get all the contents of block 1, and half the contents of block 2. But what happens if you do an ARCHIVE or DELETE, specifying 2010.200 2330 as the end point? All the records from block 1 will be retrieved and then deleted. However, block 2 cannot be deleted because it contains SMF records outside the period that you specified. This means that ARCHIVE or DELETE will neither retrieve nor delete any records from that block. Therefore, if you run a DUMP and an ARCHIVE, the output sequential data set from the ARCHIVE job might contain slightly fewer records than the output from the DUMP job.

Also, because it is not permitted to have gaps in a log stream, ARCHIVE or DELETE processing uses the log block timestamp to control which log blocks are selected and deleted. Blocks are selected by comparing the log block timestamp to the end date and time specified in the IFASMFDL statements. Once IFASMFDL reaches a log block with a timestamp higher than the specified end date and time, no more SMF records will be extracted, and no more log blocks will be deleted.

Note that, at this time, SMARTENDPOINT is not supported with ARCHIVE or DELETE processing, or with programs that use the SUBSYS=LOGR interface with the SMF-provided IFASEXIT exit, although the need for this support is a known requirement. This means that, currently, an ARCHIVE or DELETE job will read the log stream from the start, right through to the end. A job using SUBSYS=LOGR with IFASEXIT will process the log stream from the start date and time that you specify in the JCL, through to the end of the log stream.

<sup>4</sup> At the time of writing, SMARTENDPOINT is *only* supported in conjunction with the IFASMFDL DUMP option.

**Tip:** When running in dataset mode, many installations specify a large value for MAXDORM, because you want to place as much data as possible in each block. However, when running in logstream mode, the use of a smaller MAXDORM (one minute or less) is suggested. This greatly decreases the time range that will be included in each log block, thereby allowing you to specify a much smaller SMARTENDPOINT value.

Table 1-1 summarizes which SMF logstream functions are available on the various releases of z/OS, and whether they are included in that release (Base) or if an APAR is required to add that function.

*Table 1-1 SMF logstream mode enhancements availability*

SMF logstream mode function	z/OS 1.9	z/OS 1.10	z/OS 1.11	z/OS 1.12
IFASMF DL DUMP	Base	Base	Base	Base
IFASMF DL ARCHIVE	OA27037	OA27037	Base	Base
IFASMF DL DELETE	OA27037	OA27037	Base	Base
MAXDORM support	OA27037	OA27037	Base	Base
NOBUFFS support	No	No	No	Base
BUFUSEWARN support	No	No	No	Base
SMARTENDPOINT	OA31737	OA31737	OA31737	Base
SMARTENDPOINT time control	No	OA34734	OA34734	OA34734
RELATIVEBYDATE	OA27037	OA27037	Base	Base
Support for ARCHIVE or DELETE to empty log stream	No	No	OA34589	OA34589

## 1.5 Migration options

Discussion of the DUMP and ARCHIVE actions brings us to the topic of longer-term storage of SMF records. When using the SYS1.MAN data sets, SMF records are always moved into some form of sequential data set, where they are stored until they are no longer required.

When using log streams, you now have a choice. You can:

- ▶ Simply replace the SYS1.MAN data sets with one or more log streams, and then move all the records from the log streams to the same sequential data sets that you use today.
- ▶ Go to the other extreme, and keep all your SMF records in the log streams that SMF writes them to until they expire (based on the retention period that you specify for the log stream).
- ▶ Have a hybrid solution, where certain SMF record types are retained in the log streams until they expire, and others are moved to the existing sequential data sets. We think that this will be the approach taken by most installations that implement SMF logstream mode.

Which of these options is the most appropriate will vary from installation to installation, with many things to be considered. Chapter 2, “Understanding your environment” on page 27, and Chapter 3, “Migration approaches” on page 37, contain information to help you identify the criteria that will factor in your decision about the best way to proceed.

In addition to these options, you can simply carry on exactly as you are today, with writing all SMF records to SYS1.MAN data sets, and not use any of the System Logger support. At the time of writing, IBM has not issued any statements of direction or any other announcements to indicate that support for SMF dataset mode will be discontinued.

## **1.6 Migration considerations**

To help you identify how appropriate the use of SMF log streams will be for your environment, this section summarizes benefits and considerations for implementing SMF logstream mode.

### **1.6.1 Benefits of SMF log stream support**

While performance is an obvious reason to consider migrating to SMF log streams, it is only one of a number of benefits that use of the SMF logstream mode delivers compared to the use of dataset mode. This section summarizes the benefits.

#### **Performance and throughput**

The support for writing SMF records to System Logger, rather than to the SYS1.MAN data sets, was originally positioned as a way to address the performance and throughput limitations associated with the use of those VSAM data sets. In informal measurements taken by this team, there is no doubt that the throughput available when writing SMF data to System Logger is a significant improvement over what we were able to achieve when writing to the SYS1.MAN data sets.

Even on a one-for-one basis, we observed a throughput improvement of 50% when using a DASDONLY log stream compared with writing to the SYS1.MAN data sets. When you combine that with the fact that you can split your SMF record types across multiple log streams, you can see that SMF log streams address any problems or concerns that you might have with the ability of SMF to keep up with the volume of data being sent to it today. The results of our measurements are summarized in Figure 1-9 and reported in more detail in Appendix A, “Relative capacity measurements” on page 115.

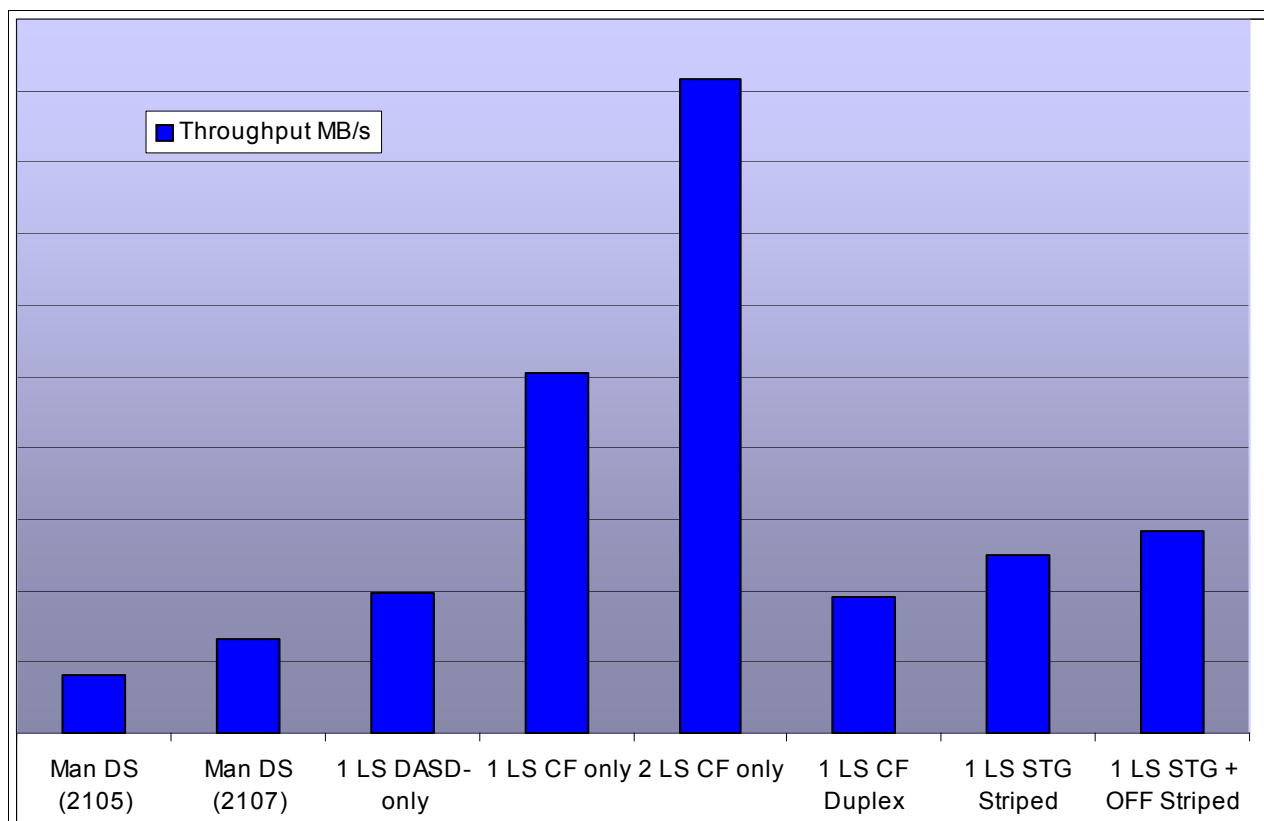


Figure 1-9 Relative throughput of SMF log streams compared to SYS1.MAN data sets

### Sysplex-wide repository

In order to obtain a complete sysplex-wide view of activity, many programs require a subset of SMF record types, but from *all* the members of the sysplex. However, the SYS1.MAN data sets contain all record types, but only for a single system. To generate a sysplex-wide report, you need to wait until all SMF data sets have switched and the record types that you require have been extracted, and then those records must be merged with the corresponding records from the other members of the sysplex.

By contrast, with SMF log streams, you have the option to have a sysplex-wide log stream, meaning that the records are written directly to a sysplex-wide repository containing the corresponding records from all members of the sysplex. This might enable you to create a sysplex-wide report with no additional pre-processing, *and* to do so as soon as the records are created.

### No-buffer condition

When SMF logstream mode was introduced in z/OS 1.9, there was no NOBUFFS or BUFUSEWARN support. While these keywords could be specified, they were ignored when SMF was running in logstream mode.



Starting with z/OS 1.12, it is possible to specify these keywords at the log stream level. This is a significant enhancement over the support that was available in SMF when running in dataset mode.

Most installations have some SMF record types that are nice to have, but it is acceptable if some get discarded. Then there are a set of record types that you would prefer not to lose if at all possible. Finally, there might be record types that are so critical that you might be willing to IPL the system rather than take the risk of losing some of those records.

In dataset mode, you can only specify a single action to be taken if the SMF buffers fill, regardless of the SMF record type that encountered the buffer-full condition. However, starting with z/OS 1.12, you can now have a much more granular approach when running in logstream mode. For the SMF record types that you absolutely must not lose, you can specify NOBUFFS(HALT) for only the log streams that contain those critical record types. For the other log streams, you can specify NOBUFFS(MSG). This means that if SMF runs out of buffers for the non-critical record types, they will be discarded. The only situation in which the system is put into a wait state is when the buffers for the critical record types fill up.

This might improve availability because the system will not be put into a wait state because you have too many records that are not critical to your installation. Because each log stream has up to 2 GB of buffers per system, the chances of running out of buffers when a critical record arrives in SMF is far less than would be the case in dataset mode (where all record types are sharing a single 1 GB buffer).

### **Quicker access**

While there are a small number of programs and products that read directly from the SYS1.MAN data sets, in general, most programs that process SMF records wait until the SYS1.MAN data set is unloaded to a sequential data set. Depending on how frequently you switch SMF data sets, this might mean waiting anywhere from a few seconds after the records are created, up to hours after they are created.

However, because SMF Logger support involves writing the SMF records directly to their “final” location (and not an intermediate location like SYS1.MAN), new SMF records are accessible as soon as they are externalized from the SMF buffers. This means that you do not need to wait for a certain event (SMF data set switch) or a given amount of time before the records can be accessed.

### **Improve efficiency**

If SMF data is stored in a sequential data set, that data must be accessed sequentially. For example, if you have a data set containing a month’s worth of performance SMF records, and you want a report for the afternoon of the last day of the month, the reporting program has to read through an entire month’s worth of records, even though it only wants five hours’ worth of records.

By comparison, when the SMF records are in a log stream, they can be accessed directly. Using the log blocks shown in Figure 1-10, let us say that you want a report for the hours of 14:00 to 18:00 (the figure only shows a subset of the log blocks in the log stream. There are both old and newer log blocks that are not shown). So in your IFASMFDDL DUMP job you specify a start time of 1400 and an end time of 1800.

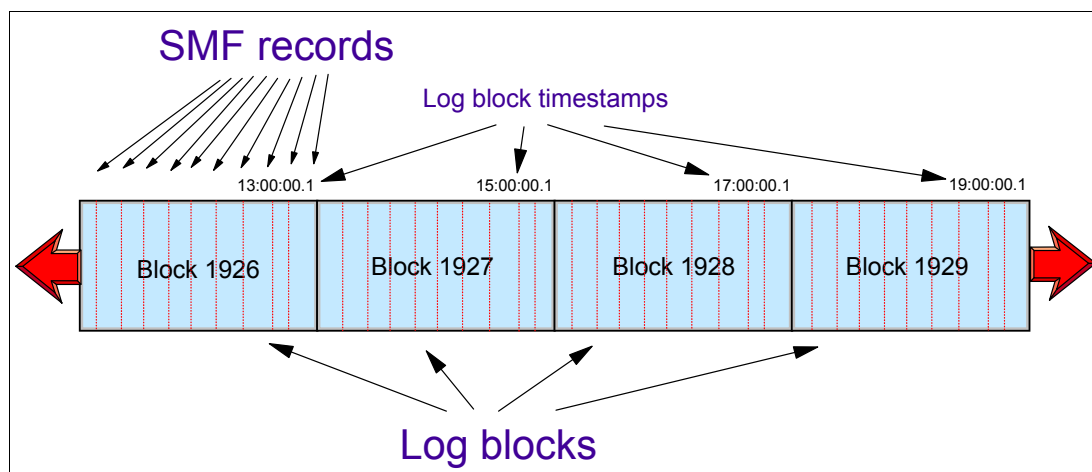


Figure 1-10 Efficient access to SMF records

IFASMFDDL takes the start time and date that you specified and passes that to System Logger, requesting the first log block after that time. So, in the example, the first log block with a log block timestamp greater than 14:00 is block 1927. Remember that the log block timestamp must always be equal to or later than the timestamp on any SMF records in the log block. So in this case, the IFASMFDDL DUMP starts with block 1927. How far it reads through the log stream depends on whether you specify SMARTENDPOINT. If you *do* specify SMARTENDPOINT, by default it reads log blocks up to two hours after the end time that you specify. So, in this example, IFASMFDDL only processes six hours' worth of SMF records, instead of a month's worth, as would be the case if the program was reading a monthly sequential SMF data set. If you do *not* specify SMARTENDPOINT, IFASMFDDL can end up processing significantly more SMF records than the job reading the monthly sequential data set processed, depending on how far back in the log stream your program needed.

### Improved availability for critical records

When SMF is using SYS1.MAN data sets, records are written to the SMF address space, where they are buffered until they can be written to the SYS1.MAN data set. They then typically remain in that data set until the data set fills and switches, at which point they are moved to a sequential data set. From the period between when the record is created and when it is moved to the sequential data set, there is no ability to have two, failure-isolated copies of those records.

Certain enterprises have identified a subset of SMF record types that are deemed to be critical to the business, and whose loss is unacceptable. If you have such records and are using SMF Logger support, you can set things up so that as soon as the records are written to SMF, they are immediately saved in two separate data spaces, then written to two log streams (located in failure-isolated CFs) that could be duplexed to failure-isolated staging data sets, and then moved to two failure-isolated offload data sets. Figure 1-11 shows such a setup.

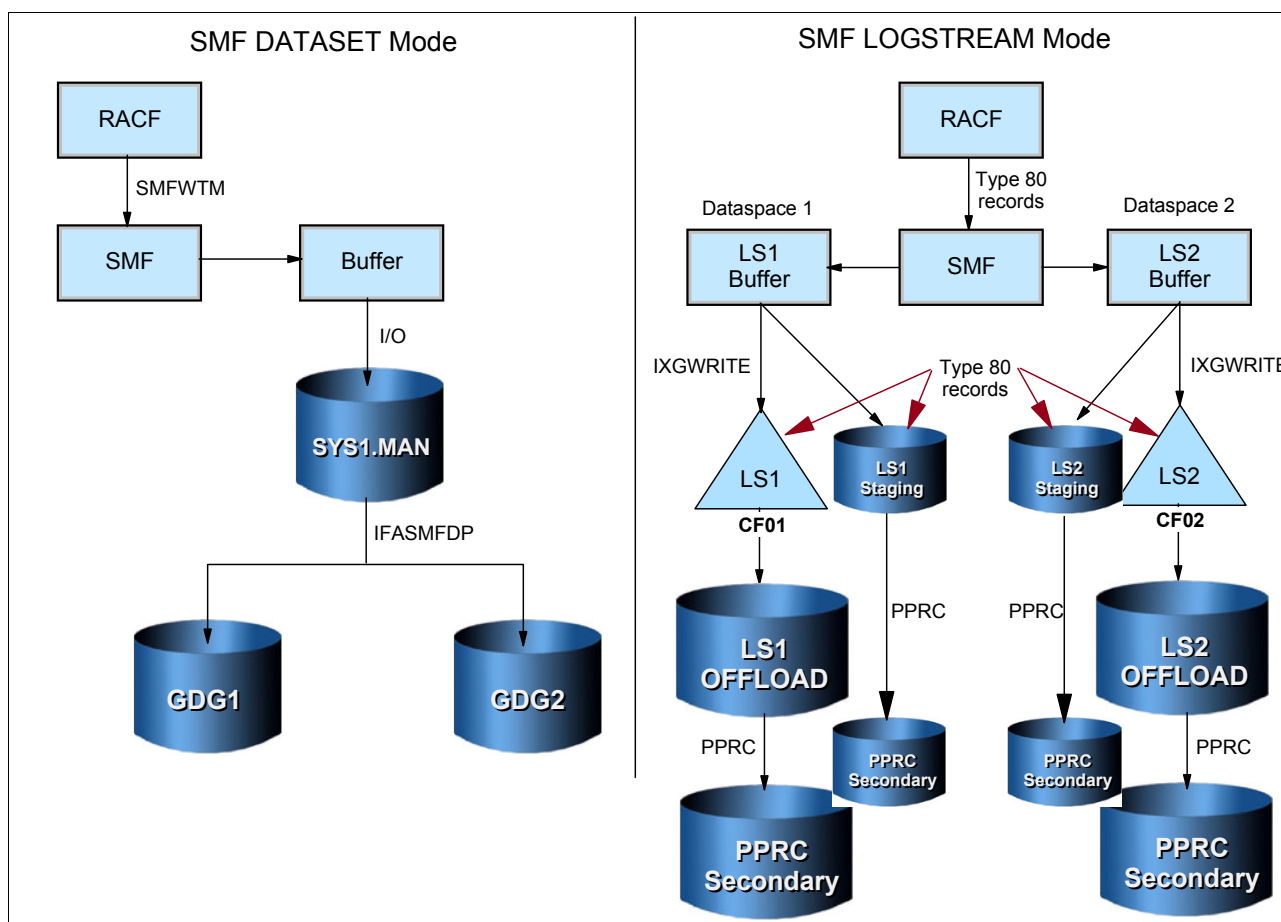


Figure 1-11 Protecting critical SMF records

### Improved protection for all record types

If you run an IFASMFDL with the ALL option against a SYS1.MAN data set (indicating that you want to retrieve the records from the data set and then clear it), all records are deleted from the file, even if your control statements resulted in just a subset of the records being moved to an output file.

However, if you run an IFASMFDL with the ARCHIVE option, and the parameters indicate that only a subset of the records should be saved to an output file, then the ARCHIVE fails. This ensures that you do not inadvertently delete SMF records that you need.

If there are records in the log stream that you want to delete without extracting them, then the most efficient way to achieve the same end result is to turn off the collection of those records in the SMFPRMxx member. If you have a reason why you *must* collect them, but you also do not want to extract them from the log stream, then you can add a DD DUMMY as the destination for those records in the IFASMFDL job.

## 1.6.2 SMF logstream mode considerations

Any time that you switch to a new way of doing something, there are going to be implementation considerations. Some of these might affect most installations, and others only a small set of them. For certain customers, the inconvenience of handling these situations might be considered trivial compared to the benefits that are achievable from using the SMF Logger support. For others, they might range from being a complete show-stopper, down to simply limiting how fully you can exploit this new capability. Nevertheless, we want you to be aware of the items that might impact your decision about whether to proceed with a migration to SMF logstream mode.

### Process changes

Most mainframe installations have an infrastructure built up around processing SMF data. Some of these might be quite simple, but others are intricate and complex. Depending on which approach you decide to take when implementing SMF Logger support, some, or maybe all, of these processes will require changes.

Chapter 2, “Understanding your environment” on page 27, discusses investigating and documenting your current SMF processes. The information you gather will then be used as input to Chapter 3, “Migration approaches” on page 37, where we discuss the options and the considerations to help you decide on the most appropriate way to implement logstream mode in your installation.

You must also plan to provide education to the operators. They will be familiar with managing the SYS1.MAN data sets, and know what type of things to check in case of problems. You will need to train them in how to achieve the same results in an environment where SMF is writing to log streams rather than the SYS1.MAN data sets. Appendix B, “Important command and message changes” on page 123, contains a summary of new and changed messages and commands of which the operators should be aware.

### Limited ability to phase in

Many installations would like to “dip a toe in the water” by moving just some SMF record types to a log stream, while continuing to use SYS1.MAN data sets for all the other record types. However, the SMF support for log streams has not been implemented in this way. A given system must either be in logstream mode (meaning that *all* record types are being written to some log stream) or dataset mode (meaning that *all* record types are being written to a SYS1.MAN data set).

It is possible to have one or more systems in the sysplex running in logstream mode, and others running in dataset mode. In fact, we expect that nearly everyone will run in this mixed mode for a time as you migrate your systems from dataset to logstream mode.

### Handling data from multiple sysplexes

Because SMF records have traditionally lived in sequential data sets, and those data sets can easily be transmitted between systems, it has always been possible to create a complete installation-wide set of data sets containing SMF records from every system in every sysplex. We are not sure why you would want to do that for *all* your SMF records, however, the capability exists if you want to do so.

If you decided that the most beneficial way for you to manage all your SMF records was to keep them in log streams, then you would lose the ability to have a single installation-wide repository of all SMF records because a log stream can only be written to and accessed by systems in one sysplex.

You can, of course, still use log streams as a logical replacement for the SYS1.MAN data sets, then move all or some SMF record types to sequential data sets, and then merge those data sets just as you do today.

## **Handling data from GDPS Control systems**

Another, similar, consideration is for GDPS/PPRC Control systems. The System Logger address space does not run on a GDPS/PPRC Control system, meaning that you have no choice but to continue to use SYS1.MAN data sets on a GDPS® Control system. On its own, this should not be an issue, because the volume of SMF records being created on a Control system is typically far below the number that can be handled by the SYS1.MAN data sets.

However, if you have a real business need to have a single, sysplex-wide, repository of SMF records, then there is no way to merge the records from the Control systems into a log stream containing SMF records from the other members of the sysplex.

This might be another example of where you would use a hybrid solution, with some SMF record types being left in log streams (these would be the ones that you do *not* need to merge with the records from the Control system), and other SMF record types (the ones that you *do* need to merge with those from the Control system) being moved to sequential data sets.

## **Programs that access the SYS1.MAN data sets directly**

You might have one or more programs that process the SYS1.MAN data sets directly. If you switch to logstream mode, those programs would need to be modified. Because the IFASEXIT program does not emulate a VSAM data set, the programs would need to be changed to use a sequential interface, which in turn could use the LOGR subsystem with IFASEXIT to access the log stream. While it is possible for a program to access a log stream directly, the format of the SMF log blocks is not a published interface, so there is no defined way for you to deblock the log blocks if the program were to retrieve them using IXGBRWSE.

Another possible alternative is to use an SMF exit to pass the records that the program might be interested in.

## **Programs that directly access the SMF sequential files**

There are a multitude of programs that read sequential data sets containing SMF records. Some of these are provided by IBM, some by other vendors, and many are customer-written.

If you identify some SMF record types that you would like to keep in a log stream until they expire, then you need to consider how the programs that currently read those record types will work with a log stream instead.

One option is to add a new IFASMFDDL DUMP step that will extract the records that you want to use from the log stream into a sequential data set, and that data set would then be passed to your program. While this involves a JCL change, it has the benefit that you can use the SMARTENDPOINT function to limit the number of log blocks that are extracted from the log stream. Another benefit is that you might be able to exploit the RELATIVEDATE function to minimize the number of JCL changes required. If you decide in favor of this option, it is a good idea to do a lot of testing to ensure that you completely understand how the RELATIVEDATE function works. For example, what end date is used when you specify RELATIVEDATE(BYDAY,30,1) and the log stream goes back more than 30 days? (In this case, it uses an end date of -29).

Another option is to simply change the DD statement that points at the sequential SMF data set so that it uses the LOGR subsystem interface with the SMF-provided IFASEXIT program instead. This allows programs to access the log stream in a manner that is transparent to the program.

One thing that you need to consider if you use this method is the volume of data that will be returned to the program. Most programs that process SMF data will typically access one generation of a GDG data set, and that generation will normally hold data for one day, one week, or maybe one month. As a result, the elapsed time to read through all that data is finite.

While the SUBSYS=LOGR interface has keywords to specify a start and end date, in fact, the IFASEXIT program currently ignores the end date, and all log blocks up to “now” will be returned to the program. However, the subsystem *does* use the start date, so only log blocks from that point forward will be returned. You also need to remember that a JCL change to update the start date would be required every time that you want to look at a different date range of data (because IFASEXIT does not support RELATIVEDATE). This interface, and the considerations for using it, are discussed in more detail in 2.1.1, “Which programs access SMF data” on page 28.

## User education

Another thing that needs to be considered is handling the transition from GDG data sets to log streams.

If it was possible to change things so that *all* of a given SMF record type are in GDGs today and in log streams tomorrow, the transition would not be so challenging. The difficulty arises from the fact that there will be a period of time when SMF records from before the cutover date will continue to reside in the GDG data sets, and SMF records from that point forward will reside in a log stream, meaning that different JCL must be used, depending on where the records are to be retrieved from.

If you have users whose primary responsibility and skills lie outside the IT area, handling the mix of log streams and GDGs could prove challenging. In Chapter 2, “Understanding your environment” on page 27, you will identify the users of the different SMF record types. That information will subsequently help you decide which record types are candidates to keep in a log stream, and which should continue to be moved to sequential data sets as they are today.

## Storage requirements

The maximum amount of real storage that you can have for SMF buffers when writing to SYS1.MAN data sets is 1 GB. This is pageable storage, residing in the SMF address space.

When using log streams, you will have a minimum of 100 buffers-worth of non-pageable DREF storage for each log stream. In addition, the buffers for each log stream can potentially grow to 2 GB.

You need to remember that it will take less time to move a buffer to a log stream than it takes to move it to a SYS1.MAN data set. So, for a given workload, the likelihood is that you will probably actually have less storage tied up for SMF buffers when in logstream mode than when in dataset mode. Nevertheless, if something happens that causes the externalizing of the buffers to be delayed, there is the potential for a lot more central storage (2 GB x the number of SMF log streams) to be tied up than when you are in dataset mode (where there is a maximum of 1 GB of buffers).

## Logger workload

The MVS System Logger was originally designed to be used by interactive applications. Therefore, one of its objectives is to deliver the fastest response possible to a request to save a block of data. Some online applications will write a log block to Logger and wait for a response from Logger indicating that the log block was successfully saved before they continue with their processing. Therefore, to avoid causing any delays to these applications, you want Logger to be as responsive as possible.

One thing that you might want to consider is the impact on Logger's responsiveness for online applications if you are also asking it to retrieve gigabytes of SMF data at the same time. In other words, you are asking Logger to act as both a realtime system and an archiving system—roles that are usually separated.

Logger has multiple tasks and can process many requests in parallel. Nevertheless, it is prudent to be careful when asking Logger to manage log streams containing hundreds of gigabytes if that data is going to be used intensively. Remember that every request to write, retrieve, or delete a log record must go through the Logger address space.

## 1.7 Is SMF logstream mode right for you

If you are reading this document, you presumably have challenges with SMF today. They might be related to concerns about SMF performance, or possibly the complexity of how you currently manage your SMF data.

At this point, you hopefully understand the differences between SMF in dataset mode and in logstream mode. As you can see, there are advantages to running SMF in logstream mode. And there might certainly be valid reasons for keeping your current GDG-based SMF infrastructure. For most customers, it is possible to take the best of both worlds. Our suggestion is that you review the points in 1.6, "Migration considerations" on page 17, and see which are attractive to you, and also see whether there are any show-stoppers that would preclude you from being able to exploit this capability.

If your decision is that exploiting System Logger support in SMF might be appropriate and beneficial for your environment, the remainder of this document will help you plan for and implement a smooth and uneventful switchover to logstream mode.







## Understanding your environment

Before you can decide on the System Management Facilities (SMF) topology that is the best fit for your environment, you must first understand how your SMF data is being used today, and by whom. This chapter helps you with the process of gathering that information.

Even if you decide not to progress with a migration to logstream mode at this time, conducting a review of your SMF management processes is a worthwhile activity and might help you improve the efficiency of your current operations.

## 2.1 Why you need to know how your SMF data is used

The mechanics of changing SMF so that it writes its records to a log stream, rather than to a Virtual Storage Access Method (VSAM) data set, are trivial. After you have done your planning and preparation, the actual switchover takes no more than a few minutes.

However, far more important than the effort involved in switching from VSAM data sets to one or more log streams is the question about what to do with your SMF data after you collect it. As discussed in 1.5, “Migration options” on page 16, you have the choice of moving your SMF data from the log stream to a GDG (in much the same way as you move it from the SYS1.MAN data sets today) and managing the data from that point forward in exactly the same way as you do today. Or you can leave the data in a log stream until the data is no longer required. You can also have a combination of these approaches, with certain record types being moved from a log stream to a GDG, and others being left in their log stream until they expire.

To understand the best option for each SMF record type, we believe that it is vital to understand how the SMF data is being managed and used today, and how acceptable or disruptive a change to that mechanism would be. This chapter helps you gather the information that you need to make these decisions. As you gather the information, you will come to understand:

- ▶ Which SMF record types are being collected today.
- ▶ Whether all the records that you are collecting are actually being used.  
Do you even have tools to process all the record types that you are collecting? There is little point in collecting an SMF record type if you do not have any means to report on the information contained in those records.
- ▶ Are there duplications in the process? For example, are there two applications that both strip out and keep the same record type?

Naturally, we suggest that you take the opportunity to optimize the current SMF processing before undertaking a migration to logstream mode. There is little point in putting time into migrating a process that is not actually required or necessary.

### 2.1.1 Which programs access SMF data

If you decide to replace your SYS1.MAN data sets with log streams, you must choose where you will keep your SMF records on a longer term basis. One of the inputs to the decision of whether it is feasible to keep a given SMF record type in a log stream is to understand what programs and processes are used to access that SMF record type today. Many programs read SMF records directly from the sequential GDGs. A small number of programs read the data directly from the SYS1.MAN VSAM data sets. And some processes use the IFASMFDP program to extract the data from either the SYS1.MAN or GDG data sets into a sequential data set, and then process that data set.

After you identify all the programs that process SMF records today, it should be possible to test whether those programs will work with the SUBSYS=LOGR interface, or if the use of the IFASMFDP program to extract the required records from the log stream would be better suited. You can easily test the programs by running them against an SMF log stream on a system programmer sandbox system.

## 2.1.2 Which people access SMF data

Another input in determining the feasibility of leaving select SMF record types in a log stream is to understand who uses those record types today. It is likely that some of the users are system programmers, DBAs, or performance analysts—people who understand JCL and should be able to handle the situation where some of the historic data is still in a GDG, and some of the data is in a log stream.

However, for users of SMF data that are not familiar with JCL, and who possibly do not even know where their JCL is stored, the situation might be different.

Consider the example of an IT accountant who uses SMF data for chargeback purposes. They are probably used to the concept that the 0 in a GDG name indicates that the data relates to this week, -1 means that it relates to last week, and so on. The JCL that they have might run against a file containing a maximum of one week's worth of data.

Now consider a case in which you decide to convert those SMF record types so that they will stay in a log stream until they expire. You are immediately faced with a number of challenges:

- ▶ The input “file” no longer contains just one week's worth of data. It contains data from the beginning of the log stream, all the way up to now. This probably means that a mechanism must be found to limit the records that are passed to the program to a given date range. There are a number of ways to achieve this, but all require a change to the current processes.
- ▶ GDG -1 will no longer mean “last week.” When you switch from using GDGs over to keeping the data in a log stream, -1 becomes frozen in time, no longer being an offset relative to today or this week. This means that users must understand that they have to use two different sets of JCL, depending on the range of data in which they are interested. For the data that still exists in GDGs, they must determine which GDG data set contains the required date ranges. This is a change to the current process where the GDG name can be specified as a relative offset from today.

To help you identify who is accessing the SMF data sets today, we have provided a sample program called SMFDSACC (described in “SMFDSACC program” on page 140) that provides not only a list of data sets and the programs that access them, but also the user ID associated with that access. SMFDSACC is one of a number of tools included in this book that are designed to assist you with the migration to SMF logstream mode. Figure 2-1 shows the flow between the tools.

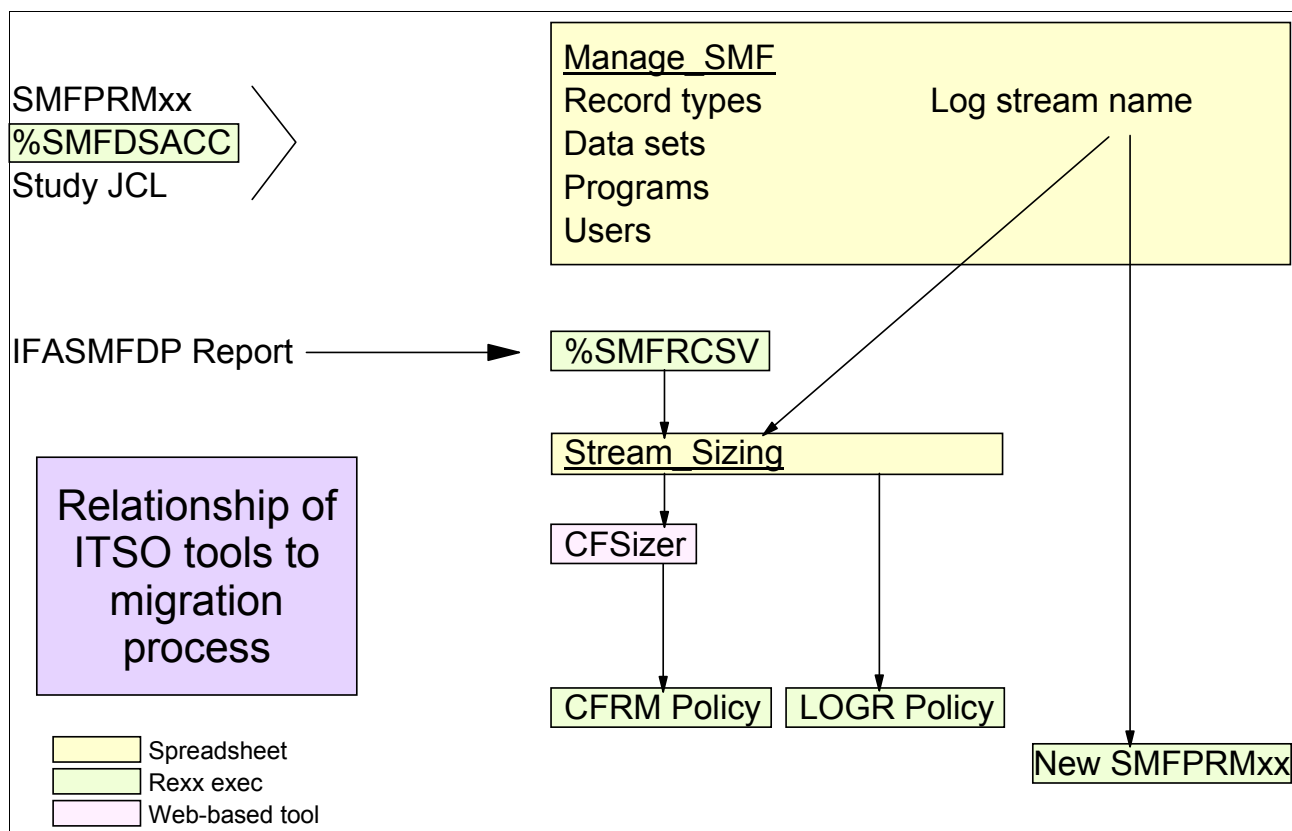


Figure 2-1 Migration tools

As you can see, the SMFDSACC REXX program creates a report that helps you complete a sample spreadsheet that we provide (called Manage\_SMF). The other execs and spreadsheets are described as we get to those parts of the migration process.

## 2.2 Documenting your current SMF processes

One of the first steps is to understand and document your current SMF processes, from the point where the record types that are to be collected are defined (in the SMFPRMxx member), through to what happens when the SYS1.MAN data sets are switched, and on to the creation of all files that contain raw SMF records.

Appendix E, “Additional material” on page 135, contains a small spreadsheet called Manage\_SMF that we hope will be helpful in your efforts to understand how your SMF data is being used today and how you might map that into your future environment. It is a repository in which to store the information that you collect as you go through the remainder of this document. The columns in that spreadsheet are described in “Manage\_SMF spreadsheet” on page 136. You might want to download the spreadsheet now so that it is available as you start gathering the information that it will contain.

## Completing the Manage\_SMF spreadsheet

This section leads you through the process of populating the Manage\_SMF spreadsheet with information about your SMF environment.

Every installation has its own SMF processes, expanded and fine-tuned over many years. A common methodology seems to be that the dumping (and clearing) of each SYS1.MAN data set on each system creates a new generation of a GDG containing all the records that were in the SYS1.MAN data set (the objective is usually to save and clear the SYS1.MAN data sets as quickly as possible). Then, shortly after midnight, a set of jobs is submitted that will process these GDGs and extract selected SMF record types for the previous day into another set of GDGs. Depending on the record type, these jobs might extract information from just one system, or they might create GDGs that contain records from the entire sysplex.

This is just one methodology. There are as many variations on this as there are z/OS customers. But the underlying principle of extracting data from the SYS1.MAN data sets and then splitting the record types out into multiple data sets appears to be common across nearly all installations.

### ***Which SMF record types are being collected***

As an initial step, we suggest issuing a **D SMF,0** command on each system to determine which SMFPRMxx member is being used on that system, and to identify which record types are (and are *not*) being collected. Use this information to complete Column C (Collected?) in the Manage\_SMF spreadsheet.

This is also a good time to complete Column M (Lost records acceptable?) and Column N (Critical records?) in the spreadsheet. “Lost records acceptable” means that it would be acceptable to lose a small number of these record types in case of a double failure affecting one or more systems and the CF containing the log stream. Critical records are record types that you absolutely must not lose. This information will be used later to identify the availability characteristics of the log stream to which each record type is assigned.

This is a good opportunity to review which records you are collecting and whether they are still required. The standard IBM SMF record numbers are in the range 0 to 127, but not all of them are created in every environment, and some of these records are obsolete because the information contained in them has been moved to other records, for example:

- ▶ Type 04 Step termination  
This information is included in the type 30 records.
- ▶ Type 05 Job termination  
This information is included in the type 30 records.
- ▶ Type 10 Allocation recovery  
The response to message IEF238D is available in Syslog.
- ▶ Type 20 Job initiation  
This information is included in the type 30 records.
- ▶ Type 34 TSO/E Step termination  
This information is included in the type 30 records.
- ▶ Type 35 TSO/E logoff  
This information is included in the type 30 records.
- ▶ Type 40 Dynamic DD  
This information is included in the type 30 records.

Ideally, the collection of these record types will be turned off (to avoid duplication), and the information that they contained will be retrieved from elsewhere if needed.

**Tip:** CICS TS 3.2 added the ability to compress the CICS data section of the CICS SMF Type 110 records. Depending on your CICS workload, enabling this capability can significantly reduce the volume of data that is written to these records. When this capability is enabled, the SMF header and SMF product sections will continue to be written in uncompressed format. As a result, the use of compression for the CICS data section of the type 110 records is supported in logstream mode just as it is in dataset mode.

### Where does the SMF data reside and who uses it

Now that you have a list of which record types you are collecting, we will gather information about how each record type is being used and managed.

The sample REXX program called SMFDSACC (documented in “SMFDSACC program” on page 140) helps with this process. The program and a sample job to run it can be downloaded from Appendix E, “Additional material” on page 135. The program reads SMF type 14, 15, and 62 records and produces the following reports:

1. The first report (Example 2-1) shows all executions of the IFASMFDP program and the input and output data sets that were used. The output data sets in this case *probably* contain SMF records (because it is possible that the SYSPRINT, for example, from the program was directed to a data set rather than a spool file).

*Example 2-1 SMFDSACC report 1*

SMF dump program executions

Day	Time	SysID	Jobname	UserID	Program	I/O data set name	Type	Rfm	Gen	Retd
243	030000	ZT01	DUMPXY	SYSSTC	IFASMFDP	SYS1.ZT01.MAN2 SMF.DUMP.ZT00PLEX	MAN GDG		VBS	250
243	065744	ZT01	LENNARTA	LENNART	IFASMFDP	SMF.DUMP.ZT00PLEX SYS2.R141562.F2	GDG SEQ		VBS	250
243	065931	ZT01	LENNARTA	LENNART	IFASMFDP	SMF.DUMP.ZT00PLEX SYS2.R141562.F2	GDG SEQ		VBS	250
243	065955	ZT01	LENNARTA	LENNART	IFASMFDP	SMF.DUMP.ZT00PLEX SYS2.R141562.F3	GDG SEQ		VBS	250
243	070018	ZT01	LENNARTA	LENNART	IFASMFDP	SMF.DUMP.ZT00PLEX SYS2.R141562.F4	GDG SEQ		VBS	250
243	074443	ZT01	DUMPXY	SYSSTC	IFASMFDP	SYS1.ZT01.MAN2 SMF.DUMP.ZT00PLEX	MAN GDG		VBS	250
243	075601	ZT01	LENNARTB	LENNART	IFASMFDP	SMF.DUMP.ZT00PLEX SYS2.SMF COPY	GDG DTG		VBS	250 1
243	075824	ZT01	DUMPXY	SYSSTC	IFASMFDP	SYS1.ZT01.MAN1 SMF.DUMP.ZT00PLEX	MAN GDG		VBS	250

The types of data sets are classified in the reports using the Type column as one of the following:

<b>MAN</b>	VSAM SMF data set
<b>GDG</b>	Data set with a name like DATASET.NAME.GnnnnV00
<b>DTG</b>	Data set with a name like DATASET.NAME.Dnnnn.Tnnnn
<b>SEQ</b>	Other sequential data sets

The Rfm column contains the RECFM for the listed data set. While it is *possible* for data sets containing SMF records to have a RECFM of FB, VB, or VBS, in general, they have a RECFM of VB or VBS, so the intent of the Rfm column is to give you a hint of whether this data set is likely to contain SMF records.

The Gen column indicates the number of generations that are defined if this is a GDG data set.

The Retd column displays, for data sets that were defined with a retention period or expiration date, the number of days (from the current day) left before the data set expires.

**Note:** The user ID information in the reports is taken from SMF field SMFxxUID. This field is blank if SMF exit IEFUJI is not activated.

This list of data sets is saved and used as input for the third report.

2. The second report (Example 2-2) is for other programs that have opened the SYS1.MAN data sets. The data sets created by these program are used as input to the third report.

Example 2-2 SMFDSACC report 2

Other jobs/programs accessing SMF MAN data sets

Day	Time	SysID	Jobname	UserID	Program	I/O data set name	Type	Rfm	Gen	Retd
190	151923	ZT01	SMF	SYSSTC		SYS1.ZT01.MAN2	MAN			
243	074239	ZT01	LENNARTQ	LENNART		SYS1.ZT01.MAN1	MAN			
					SMFDS1	LENNART.TEST.SMFDS1	SEQ	VB		116

**Important:** If you migrate to SMF log streams, the programs identified in this step will need to be modified to retrieve their data from a log stream or sequential data set rather than the SYS1.MAN VSAM data sets, or to use an SMF exit to retrieve the data in that way.

3. The third report (Example 2-3) contains a list of all jobs that used the data sets identified in the first two reports as input.

*Example 2-3 SMFDSACC report 3*

Jobs using SMF data

Day	Time	SysID	Jobname	UserID	Program	I/O data set name	Type	Rfm	Gen	Retd
243	072846	ZT01	TDSSMFC1	LENNART	DRLPLC	SMF.DUMP.ZT00PLEX	GDG	VBS	250	
243	074239	ZT01	LENNARTQ	LENNART	IEBGENER	LENNART.TEST.SMFDS1	SEQ	VB		116
243	065222	ZT01	LENNART		IKJEFT01	LENNART.SMFCOPY	DTG	VB	2	
243	080252	ZT01	LENNARTA	LENNART	IKJEFT01	SMF.DUMP.ZT00PLEX	GDG	VB	250	
243	080805	ZT01	LENNARTA	LENNART	IKJEFT01	SMF.DUMP.ZT00PLEX	GDG	VB	250	
243	081146	ZT01	LENNARTA	LENNART	IKJEFT01	SMF.DUMP.ZT00PLEX	GDG	VB	250	
243	081916	ZT01	LENNARTV	LENNART	VBSTOV	SYS2.R141562.F4	SEQ	VBS		

Between the three reports, you should have a list of all the data sets that contain SMF records and the programs, jobs, and user IDs that processed those data sets.

Unfortunately, it is not possible for this program to identify which SMF record types reside in each of these data sets. That information comes from studying the jobs that are listed in this report. While you are studying those jobs, also take note of whether the output data sets contain SMF records from a single system or multiple systems.

You can use this information to start populating the Manage\_SMF spreadsheet. The information should allow you to fill in Column D (SEQ DSNs), Column E (Accessing Programs), Column F (Accessing Jobs), Column G (Accessing Userids), Column I (Retention Period), and Column K (Sysplex or System Level). For Column I (Retention Period), you will need to combine the values from the SMFDSACC report with information obtained elsewhere (such as how often is a new generation created, what retention period was specified when the data set was created).

To get as comprehensive a list as possible, we suggest that you run the program against at least one full month's worth of SMF data, preferably from a quarter-year, half-year, or year-end period.

Having identified which SMF record types are in each data set, you are now in a position to identify which record types are grouped together into the same sequential data sets today. This is important information because:

- ▶ The grouping has probably been fine-tuned over a number of years, and therefore is a valuable reflection of how the data is used.
- ▶ The amount of time the data is kept for is nearly always determined by the number of generations in the GDG that it resides in, and how frequently new generations are created. You will use this information later to help you size the log streams.



- ▶ It indicates who has access to which record types. For example, the security violation-related records are probably grouped into one GDG, and the group with access to that GDG is probably the IT security or audit group.
- ▶ By understanding which data set a particular record type resides in, you can then document which programs are used to access that record type.

If you have been completing the Manage\_SMF spreadsheet, you can use the sort and filter capabilities to quickly see which record types reside together in which data sets. This information is used in 3.3, “How many SMF log streams to have” on page 45, to help you determine how many SMF log streams you will have.





## Migration approaches

There are a number of approaches that can be taken for managing your System Management Facilities (SMF) data into the future. This chapter helps you understand the options and determine the most appropriate approach for your environment.

**Note:** The suggestions and descriptions in this chapter assume that the PTF for APAR OA34589 is applied.

If this PTF is not applied, refer to Appendix C, “Considerations for systems without APAR OA34589” on page 131, for information about how this might affect your implementation plans.

## 3.1 Introduction

Having read Chapter 2, “Understanding your environment” on page 27, and populated the information about your SMF infrastructure into the Manage\_SMF spreadsheet, you should now have a good understanding of how you are managing and using your SMF data today. Most importantly, you will have the following information:

- ▶ What started tasks or jobs are used to empty the SYS1.MAN data sets, how those jobs or tasks are kicked off, and which data sets those jobs create
- ▶ How your SMF records are grouped (that is, which record types are stored in which sequential data sets)
- ▶ Which SMF record types are stored in data sets containing data from just one system, and which are stored with data from all systems in the sysplex
- ▶ What programs are used to read SMF records
- ▶ Which people run jobs that process SMF records

With this information, you are now in a position to investigate the options that are available to you and to map out how you will manage your SMF data as you move into the future. We believe that this is the most time-consuming part of this project.

When undertaking a significant project such as moving to a new method of managing SMF data, thorough planning is required to ensure a seamless and uneventful migration. The decisions that you make as you progress through this chapter will act as inputs to the planning chapter, and therefore play a key role in the success of the project.

### General migration considerations

Most of the approaches that we discuss in this chapter involve placing SMF data in log streams, which means that you will need to use the IFASMF DL program to extract and manage the SMF data. There are differences between IFASMF DL and IFASMF DP that will affect your plans, so it is important to understand those differences before we proceed. All of these considerations only apply to the use of log streams. If your strategy for a particular record type is to keep those records in a sequential data set, then the processing of those records (after you archive them to the sequential data set) will be unaffected.

#### *Time range of input files*

Because the input “file” to the IFASMF DL program might potentially contain a much larger time range than the input to an IFASMF DP program (which is typically one SYS1.MAN data set or one generation of a GDG), you might need a mechanism to control the time range of SMF records that will be extracted.

If your strategy for the log stream is to archive all the records to a sequential data set, then it should not be necessary to specify a start and end time (assuming that you are archiving the records on a regular basis).

However, if your strategy is to keep records in the log stream until they expire, the log stream might contain a large time range of records, so specify a start and end date (or use the RELATIVEDATE keyword), together with the SMARTENDPOINT option, to avoid the processing program being passed more data than necessary.

#### *Starting point of IFASMF DL processing*

Another important point to remember is that an IFASMF DL ARCHIVE or DELETE operation *always* begins at the start of the log stream, regardless of any start date that might be specified in the JCL. DUMP processing, on the other hand, uses whatever DATE and START

parameters you provide to control where processing of the log stream starts. As a result, if you run an ARCHIVE and a DUMP with exactly the same parameters, it is possible that the ARCHIVE will move more data because it will go all the way back to the start of the log stream, whereas the DUMP will start at whatever date and time you specify.

Also, at the time of writing, if the IFASMF DL ARCHIVE or DELETE step needs to go back further than the start time and date that you specified in order to reach the start of the log stream, it does not issue a message to tell you that the start time that you specified was ignored.

### ***ARCHIVE must move all records within specified time range***

Recall that it is not possible to delete parts of a log block. Either the entire log block is deleted from the log stream or none of it is. It is also not possible to delete one log block but *not* to delete an older one. For this reason, every SMF record in the specified time range for an ARCHIVE operation *must* be moved to an OUTDD data set. If any record is encountered that does not match the selection criteria on any of the OUTDD statements (meaning that it would not be moved to any of the OUTDD data sets), the offload will fail.

As an example, assume that you have a log stream containing SMF records from two systems and that you want to move the records to two sequential data sets, one for each system. If you try to do this as two separate ARCHIVE steps (one for each system), the ARCHIVE fails (because each attempts to move just a subset of the records in the log stream). Instead, run a single ARCHIVE step that moves the SMF records to one sequential data set, and then split the records from there out to the two sequential data sets (using a program like ICETOOL).

Note that it is possible that SMF records have already been copied to one or more output files at the point that IFASMF DL fails because it encountered a record that does not match any of the selection criteria. In this situation, you might want to subsequently run a program against the output data sets to remove duplicate records. A sample job to perform this processing is described in “DELDUPS job” on page 144.

### ***Minimizing JCL or parameter changes***

The RELATIVEDATE function (available with z/OS 1.11 and rolled back to z/OS 1.9 with APAR OA27037) allows you to select a range of records relative to the current day, week or month. This parameter gives you the ability to specify the unit of time (DAY, WEEK, or MONTH), how many units back from today to start, and the number of units to gather. So, for example, RELATIVEDATE(BYWEEK,2,1) moves back two weeks and gathers one week's worth of data. (The day that you use to signify the start of a week defaults to Sunday, but can be overridden using the WEEKSTART keyword in the IFASMF DL control statements.)

If you plan to move SMF records from the log stream on an hourly basis (for example), we suggest that you specify RELATIVEDATE(BYDAY,0,1). This specifies to move all data from the log stream, ending at the time that the program started running. This can help avoid a situation in which the IFASMF DL job runs on and on, constantly chasing the end of the log stream.

### ***Differences between ARCHIVE/DELETE and DUMP processing***

Finally, you need to be careful about differences between how DUMP and ARCHIVE/DELETE select records from the log stream.

When you run IFASMF DL with the DUMP option, the program looks at the SMF records inside every log block and selects the records that match your filtering statements.

However, ARCHIVE and DELETE work slightly differently. Remember that as part of their processing, ARCHIVE and DELETE will delete log blocks from the log stream, so ARCHIVE

and DELETE work at the log block level, rather than the SMF record level. So, the timestamp that you pass to ARCHIVE or DELETE will be used to select *log blocks* based on the log block timestamp, whereas the timestamps that you pass to DUMP are used to select *SMF records* based on the timestamps in the SMF records (because you can extract records from a log block as long as you do not try to delete half of the log block). This means that a DUMP followed by a DELETE might return slightly different results than an ARCHIVE.

**Tip:** Specifying a small MAXDORM value in your SMFPRMxx member reduces this effect by reducing the time range of SMF records that exist in a single log block.

### **Data set switching**

In the past, whenever one of the SYS1.MAN data sets became full, SMF automatically switched to the next empty data set. The entire switched-from data set was then emptied and the contents moved to some form of sequential data set.

With log stream processing, there is no switch of SMF data sets (because the log streams are a continuous stream of data), meaning that the jobs that offload the SMF records will not be triggered automatically. Depending on how you decide to manage your SMF records, you might or might not need a replacement way to kick off a process to move SMF records to sequential data sets. If your strategy *is* to move some or all record types to sequential data sets, the options are:

- ▶ Add a timed command to your automation product to kick off a started task or job that will ARCHIVE SMF records from the log stream.
- ▶ Add a timed command to your automation product to issue an **I SMF** command on a regular basis (every hour, for example). This results in the IEFU29L exit<sup>1</sup> (the logstream mode equivalent of the IEFU29 exit) being called, and you can use that to initiate a job or started task, just as you might do with the IEFU29 exit in SMF dataset mode.

This method has the advantage that SMF will close all its buffers and force them to be written to System Logger. If you have reporting processes that are based on calendar days, doing an **I SMF** just after midnight ensures that all SMF records pertaining to the previous day are available in the log streams.

- ▶ You can add an application to your batch scheduling process that will submit a batch job on a regular basis to run an ARCHIVE against the log stream.

## **3.2 Migration options**

In this section we describe the approaches that you can take. After we have described the available options, in 3.3, “How many SMF log streams to have” on page 45, we discuss the attributes of an SMF record and how it is used that are relevant to your decision about the best way to manage each record type.

Depending on your SMF environment, your requirements, and whether you are running a base sysplex (one with no CF) or a Parallel Sysplex environment, certain approaches might not be feasible or available. If you choose to implement SMF logstream mode, you need to do the transition from SYS1.MAN data sets to log streams on a complete system-by-system basis. It is not possible on a single system to have SMF writing to both types of medium at one time, although you can dynamically switch back and forth between one mode and the other.

---

<sup>1</sup> A sample IEFU29L exit is described in “Implementing the sample IEFU29L exit” on page 145.

### 3.2.1 Approach 1: No change

The good news is that if you are happy with today's SYS1.MAN processing and have no issues with the volume and rate of SMF data recording, you can continue using SMF Virtual Storage Access Method (VSAM) data sets as you do today. The new support for SMF use of log streams is entirely optional. IBM has not made any announcements or statements of direction to indicate that support for the traditional SYS1.MAN data sets will be discontinued.

If you *do* have intermittent issues with losing SMF records because of the volume of records being produced, but you want to continue using the SYS1.MAN data sets, then your options are somewhat limited.

SMF does not support the use of extended format data sets for its SYS1.MAN data sets. This means that you cannot benefit from data set striping (which supports more concurrent I/Os) or VSAM data sets larger than 2 GB. Those options for improving the performance of the SMF data sets are not available to you.

Ensure that the SYS1.MAN data sets are on the fastest DASD that you have. To give you an indication of the difference that different DASD technologies can make, we ran an informal measurement with the SYS1.MAN data sets on 2105-F20 devices and another measurement using 2107-922 devices. For both measurements, we generated a volume of SMF records to saturate the device containing the SYS1.MAN data set. When the data set switched, we then checked to see how much data was in the data set, and how long it took to fill the data set. This allowed us to calculate the maximum rate of SMF records that the device could keep up with.

In this informal measurement, the 2107-922 DASD was able to handle a data rate nearly 60% higher than the 2105 was able to handle. While the results are related to the amount of cache in the control units, the type and number of channels that were used to connect the devices, and other variables, the results *do* illustrate how much of a difference the DASD technology can make.

If you have done all that you can in terms of using the fastest available DASD, you are using the maximum SMF buffer size, and you have verified that the SYS1.MAN data sets are defined in line with IBM's guidance<sup>2</sup>, then the only option left is to start disabling select SMF record types in an attempt to reduce the volume of SMF records to a level that your DASD can keep up with.

The advantage of this approach is that no changes are required to your existing SMF processes.

---

<sup>2</sup> The IBM guidance for the SYS1.MAN data sets is documented in the section titled "Using DEFINE to Create SMF Data Sets" in *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

### 3.2.2 Approach 2: Replace only the SYS1.MAN data sets with log streams

In this approach, you simply replace your SYS1.MAN data sets with one or more log streams, but all your processes after the point where you empty your SYS1.MAN data sets do not change. Figure 3-1 shows this. The objective of this migration path is to exploit the performance and throughput improvements that are available when writing SMF records to log streams, while at the same time minimizing the number of changes to your existing SMF processes.

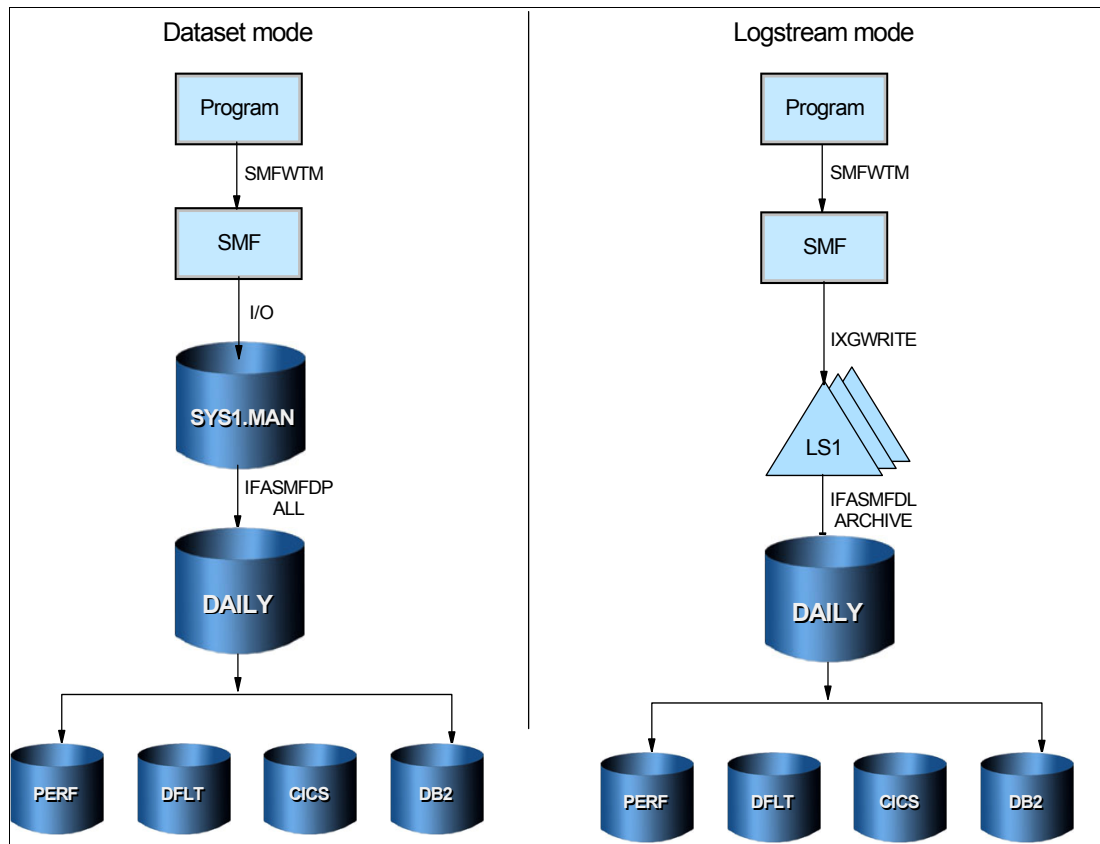


Figure 3-1 Approach 2

If you select this approach, conceptually it does not make any difference if you replace the SYS1.MAN data set with a single log stream or multiple log streams. In either case, the SMF data only resides in the log stream for a short time, just as your SMF data today only exists in the SYS1.MAN data sets for a reasonably short time. This approach pre-supposes that all the data in the log streams will be archived to the sequential data sets on a regular basis (probably hourly, but certainly no less frequently than once a day).

You will need to decide how many log streams you want, what type of log stream they should be (DASDONLY, CF and staging data set, or CF-only), and whether the log stream should contain SMF data from just one system or multiple systems. However, we suggest that you do not make this decision yet, but wait until you review the other approaches and have decided on your eventual SMF topology.

### 3.2.3 Approach 3: Keep certain SMF records in log streams

Conceptually, the simplest way to manage all SMF data is to write SMF records to a set of sysplex-wide log streams and leave all the data there until it reaches its expiry date, at which



point Logger deletes it automatically. This would make it much easier to access whatever SMF records you need. As long as you know the log stream name, you do not need to be concerned about which GDG generation contains the time period you are interested in, and you avoid the cost of moving the SMF records multiple times before they reach their final home.

However, the realities and constraints of your current SMF infrastructure need to be factored into your decision about how to proceed. Certain record types are good candidates for leaving in a log stream until they expire, but others might not be.

The SMF record types that are good candidates for being left in a log stream until they expire typically meet most of the following criteria:

- They are accessed by a limited number of programs and a limited number of jobs.

There is a simple reason for this suggestion: For every record type that you decide to keep in a log stream, JCL changes are required to every job that processes those records. The more jobs and programs there are that process a given record type, the more work is involved in migrating that record type to a log stream.

There are two options for handling programs that currently process SMF records from sequential data sets:

- You can add an IFASMFDDL step to extract the required data from a log stream into a temporary sequential data set and pass that data set to your program.
- You can use the SUBSYS=LOGR interface with the SMF-provided IFASEXIT to let your program read directly from a log stream.

Regardless of which option you select, JCL changes *will* be required to the jobs that access that data.

- Generally, most of the processing is done against recent records rather than ones from long in the past.

If you use IFASEXIT to let your program access the log stream, more data might be passed to your program than you expect because IFASEXIT currently does not support SMARTENDPOINT. This is likely to be more of an issue if you regularly access historical data (that is, SMF records from weeks or months ago).

You can, of course, insert an IFASMFDDL DUMP step in such jobs, and then use the SMARTENDPOINT option to limit the time range of data that is returned to your program and make that processing more efficient.

- The records are not retained for a long time.

It is simpler if you can avoid having to run for a long time with some data being in the old sequential data sets, and some being in the log stream. The reason for this is that as long as a given record type exists in both GDGs and a log stream, users need to be aware of this and make a conscious decision about which source they need to run against and which set of JCLs to use. By selecting record types that are not kept for long, you minimize the amount of time when users will need to access both media.

- You have control over the programs and JCL that are used to access them.

Remember that JCL changes *will* be required to get data from the log stream instead of sequential data sets, so you want to be sure that the JCL is stored somewhere that you can change it.

Similarly, if it transpires that any program changes are required, it is necessary for you to have access to the program source.

- The users of the records are reasonably IT-literate.

Because there will be a period when the more recent records are in log streams and the older records are still in sequential data sets, users of the records will need to understand which repository they need to use to obtain a given set of data and how that affects the JCL that they will have to use.

- The records are only *accessed* from one sysplex.

Remember that a log stream is only accessible from systems in the same sysplex. While IBM recommends that you do not access data sets belonging to one sysplex from a system in another sysplex, it *is* possible to do so when using sequential data sets.

- The current repository does not *contain* records from more than one sysplex.

While it is possible to merge SMF records from any number of systems in any number of sysplexes into a sequential data set, it is not possible (or at least, not feasible) to merge records from multiple sysplexes into a single log stream.

- The current repository does not contain records from a GDPS/PPRC Controlling system along with the corresponding records from the other members of the sysplex.

For reasons specific to GDPS/PPRC, the System Logger address space does not run on a GDPS/PPRC Controlling system. This means that you have no choice but to continue to use SYS1.MAN data sets on the Controlling system.

While many installations merge their Controlling system SMF records with the SMF records from the other systems in the sysplex, it is possible that this is being done because this is the way we have always done it. Given that the only thing that should run on a GDPS Controlling system is GDPS, and it should not use resources used by the other members of the sysplex, you might find that there are not many SMF records from the Controlling system that you actually use. However, if you have a genuine need to store some GDPS Controlling system SMF records in the same repository as the corresponding records from the rest of the sysplex, then that repository cannot be a log stream.

This approach naturally infers that you will have more than one SMF log stream, because different record types will have different retention requirements. In 3.3, “How many SMF log streams to have” on page 45, we discuss the process of identifying which SMF record types are good candidates to place in the same log stream. In 4.2.2, “Which type of log stream to use” on page 50, we help you determine the most appropriate type of log stream for a given record type.

### 3.2.4 Approach 4: Eliminate SMF GDGs

The final approach is to replace the SYS1.MAN data sets with log streams and then leave *all* the SMF record types in log streams until they expire.

While this is conceptually simpler than approach 2 or approach 3, because all SMF data will be treated in the same way, the reality is that this approach is more likely to be achievable in installations that do not do a lot of post-processing of SMF records, or installations with relatively small volumes of SMF data. It might also be appropriate for new z/OS installations.

### 3.2.5 Migration sequence

Most installations will have a number of objectives if you decide to implement SMF logstream mode, such as:

- You want to realize the performance benefits of log stream mode.
- You want the transition to be as uneventful and transparent as possible.

- You want to minimize the number of changes that must be made.
- You want to end up with an SMF infrastructure that is manageable, scalable, and easy to use and understand.

Given these objectives, we believe that a sensible methodology is to migrate to logstream mode in a phased manner. The first phase is to implement approach 2 across all systems in the sysplex on a system-by-system basis.

For the sake of simplicity, you probably want to implement approach 2 on all systems in the sysplex before moving on to approach 3. The reason for doing this in this manner is that you probably do not want to go through a period where you say “if you are looking for record type X for August 2, and it is from SYSA, then it is in a log stream. But if the records are from SYSB for the same date, then they are still in a data set.” By implementing approach 2 on all systems, the SMF records for all systems continue to end up in sequential data sets, avoiding this complexity.

Then, if you decide that the optimum SMF infrastructure for you is the one described in approach 3 or approach 4, it is easy to change the repository for a given record type at the sysplex level. You simply stop archiving those SMF records to the sequential data sets and change the JCL for any jobs that previously read those data sets to extract the records from the log stream instead. This enables you to make a single change that will change the repository for a given record type across the entire sysplex level.

**Tip:** Given the way that most installations manage their SMF data, we believe that the least disruptive migration path is following approach 2 for all systems in the sysplex as a first phase of the implementation. At the end of this phase, all SMF records in the sysplex are initially being placed in one or more log streams, and all of them are then being extracted to sequential data sets. This permits you to continue to use your existing SMF processes and reporting as they are today, with no changes. It also provides for easier fall-back in case you want to move back to dataset mode for some reason.

Should you decide that you then want to move a step further and keep certain records in a log stream and dispense with the sequential data sets, you can implement that change on a sysplex-wide basis. This means that prior to this switch, *all* the selected record types for the sysplex reside together in a sequential data set, and after the switch, *all* the new records of that type for the sysplex reside together in one log stream.

### 3.3 How many SMF log streams to have

There are a number of reasons why you are likely to end up splitting your SMF record types across a number of log streams:

- Look at your current implementation. If you split your SMF records across a number of sequential data sets today, there is probably a good reason why you have done it that way. The logic that resulted in your current sequential data set configuration is also applicable to a logstream environment.
- Using multiple log streams provides more throughput capacity than a single log stream.

If you follow the approach of replacing the SYS1.MAN data sets with just one log stream, then you will almost certainly want to use a staging data with that log stream to ensure the availability of your critical records. While this is a logically simpler approach, the throughput improvements will be constrained by the fact that every log block write still requires an I/O to the staging data set, and all record types are still being sent to just one destination.

If you are not constrained by the performance of your SYS1.MAN data sets today, this is a viable approach. However, if you are considering the use of SMF log streams because you are already producing larger volumes of SMF records than your DASD can keep up with, you probably want to consider moving to more than one log stream.

While the discussion about whether to use a DASDONLY or a CF log stream is an important one, at a high level, the location of the log blocks is transparent to the processing and accessing of that data, so we postpone that decision until 4.2.2, “Which type of log stream to use” on page 50.

- ▶ Keeping in mind that you cannot delete part of a log block or random blocks in a log stream, all the record types in a given log stream should be managed in the same way. That is, either they should all be moved to sequential data sets (and deleted from the log stream), or they should all be retained in the log stream for the same amount of time. It does not make sense to mix records kept for seven days with records kept for one year in the same log stream. There is only one retention period defined for a log stream, so you would end up with *all* the records in the log stream being kept for one year in that case.

If your plan for a given log stream is to move the records to sequential data sets, the ARCHIVE job must be set up to ensure that every possible record in the log stream will match one of the OUTDD statements. For example, you cannot run one ARCHIVE step to extract the records for SYSA, followed by another ARCHIVE step to extract the records for SYSB. The first step must move the records for *all* systems and *all* record types in the log stream.

- ▶ As discussed previously, some record types might be critical to you, meaning that the log stream that they are stored in should use a staging data. You might even set up two failure-isolated log streams to hold these record types. You could also specify that the system should enter a wait state rather than discard any of these records (whereas for all other record types, it is acceptable to discard them if no SMF buffers are available) by specifying a different NOBUFFS value for the log streams that contain the critical record types.
- ▶ If you have record types that are not critical, those record types can use a CF-only log stream and benefit from the significantly higher throughput that is available with this type of log stream.
- ▶ For access-control reasons, you might want to place certain record types in a log stream to which only selected users have access. Remember that access control is at the log stream level, not at the record type level.
- ▶ For performance reasons, it is preferable to avoid large Logger CF structures (more than 500 MB). The size of a Logger structure is related to the write rate, so having a small number of log streams can result in structures that are larger than is desirable. The CFSizer tool helps you size Logger structures based on the write rate.

One of the key inputs to your decision about how many log streams to have is how your SMF record types are spread across sequential data sets today. Examples of how SMF record types are grouped are:

- ▶ Performance SMF records (types 16, 50, 70 to 79, 88, 99, and 113)
- ▶ Security records (types 80 to 83)
- ▶ CICS (type 110)
- ▶ DB2® (types 100 to 102)
- ▶ Storage management (type 42)
- ▶ Chargeback (type 30)
- ▶ Catalog recovery (types 60, 61, 65, and 66)
- ▶ Software charging (70 and 89)
- ▶ MQ (type 115 and 116)

- ▶ WebSphere® Application Server (type 120)
- ▶ Everything else

The way that you group record types today is a valuable starting point for how you might group the record types in log streams.

**Tip:** While you have complete flexibility about which log streams you use, we do have two specific suggestions:

- ▶ Because the SubCapacity Reporting Tool requires SMF records from *every* system in your environment (across all sysplexes), we suggest that you continue to extract the SMF Type 70 and Type 89 records into sequential data sets.
- ▶ If you have a product that uses SMF records to recover damaged catalogs, we suggest that those SMF records are extracted from the log streams into *two* sets of sequential data sets, and that those data sets have different high-level qualifiers and that those high-level qualifiers are cataloged in different catalogs. This should ensure that you can easily access the required SMF records no matter which catalog has been damaged.

One thing to keep in mind is that the use of multiple log streams requires slightly more significant changes to your current SMF processes because the single job or STC that empties the SYS1.MAN data sets today will most likely be replaced with a number of equivalent jobs, one for each log stream. (While you can process multiple log streams in a single invocation of IFASMF DL, the overall elapsed time is reduced if you run a separate job for each log stream.)

Right now, your primary concern is to map out how you will bridge from the current environment, where each system writes to its own SYS1.MAN data set, to your target environment, where each system writes to log streams, and some of those log streams are regularly emptied into sequential data sets. Aim to create the log stream layout that you plan to end up with, and use that layout from the beginning, with more systems writing to the log streams as you convert each system over to logstream mode.

In Chapter 2, “Understanding your environment” on page 27, you populated the Manage\_SMF spreadsheet with information about how each of the SMF record types are currently being managed. Using that information, together with the considerations listed here, now go into that spreadsheet and assign record types to log streams. As you go through this process for each record type, complete column P (Log stream name) in the Manage\_SMF spreadsheet. This information is then used as input to another spreadsheet that we discuss in Chapter 4, “Planning” on page 49.





## Planning

This chapter describes the steps involved in planning for your migration to System Management Facilities (SMF) logstream mode.

If you have reached this far in this book, you have decided that you want to proceed with the migration to logstream mode. This chapter uses the information that you have gathered thus far to bring you to the point where you are ready for the cutover to logstream mode.

## 4.1 Layout of this chapter

We believe that most installations will follow either approach 2 or approach 3, as described in Chapter 3, “Migration approaches” on page 37. Even if you decide to use approach 4, that is basically just an extension of approach 3. Therefore, this chapter provides information in the following sequence:

- ▶ Planning steps that apply to *any* sysplex that will implement SMF log stream mode.
- ▶ The steps to prepare for approach 2. That is, the SYS1.MAN data sets will be replaced with one or more log streams, and all SMF record types will then be archived from those log streams to the existing sequential data sets.
- ▶ The steps to move from approach 2 to approach 3 (or approach 4). That is, you now want to leave some record types in the log stream rather than moving them to sequential data sets.

## 4.2 Preparing for SMF logstream mode

This section describes the steps that apply to any installation that plans to implement SMF logstream mode, regardless of which approach you intend to follow.

### 4.2.1 How many log streams to have

One of the fundamental decisions that you need to make before you proceed with your implementation is how many log streams you will have and which SMF record types will be assigned to each log stream.

This topic was discussed in 3.3, “How many SMF log streams to have” on page 45, and the Manage\_SMF spreadsheet should at this point contain the name of the log stream that will be used for each SMF record type.

**Tip:** While low activity LPARS such as system programmer sandboxes, GDPS/XRC Control systems, and some test or development LPARS would be fine storing all their SMF data in a single log stream, we believe that it is best to use the same SMF configuration for all your LPARS. So, if your optimal configuration for the production sysplex is eight log streams, we suggest that you also use eight log streams in the other sysplexes. The log stream sizes (structures, staging, and offload data sets) can be smaller, but the topology should be the same.

### 4.2.2 Which type of log stream to use

System Logger always keeps two copies of log blocks between the time when they are written to a log stream and when they are moved to an offload data set. The choices are:

- ▶ One copy in a System Logger data space in z/OS, and the second copy in a staging data set on DASD. This is known as a DASDONLY log stream.
- ▶ One copy in a System Logger data space in z/OS, and the second copy in a CF structure. This is known as a CF-only log stream.
- ▶ One copy in a CF structure, and the second copy in a staging data set on DASD. We call this a CF+staging log stream.



The throughput of both DASDONLY and CF+staging log streams is limited by the fact that every write to that log stream requires an I/O to a staging data set. CF-only log streams typically have significantly larger throughput capacity.

From the perspective of the program writing to the log stream, there is no difference between the three types of log stream. However, there are a number of differences when looked at from the perspective of a user of the log streams:

- ▶ A DASDONLY log stream can only contain data from one system in the sysplex. If the ability to produce sysplex-wide reports from a single log stream is attractive, then you must use a CF-only or a CF+staging log stream.
- ▶ A DASDONLY log stream can only be accessed by one system at a time. This means that any jobs that retrieve data from that log stream must run on the system that is currently using that log stream. On the other hand, a CF+staging or CF-only log stream can be accessed from multiple systems in the sysplex concurrently.
- ▶ Both a DASDONLY log stream and a CF+staging log stream protect the data in the log stream from a failure that might impact both the Coupling Facility *and* the connected z/OS system.
- ▶ All three types of log streams support the ability to extract SMF records for just one system should you so desire.
- ▶ Both a CF+staging and a CF-only log stream use some capacity in the Coupling Facility. DASDONLY log streams do not use any CF capacity.

On balance, we believe that the best option is to use either CF-only or CF+staging log streams. These have the benefit of providing significantly more flexibility. They can be accessed by jobs running anywhere in the sysplex, and they provide the ability to run sysplex-wide reports from a single repository. If you want to continue to process records on a system (rather than a sysplex) basis, you can do that as well. In other words, either type of CF log stream can do anything that a DASDONLY log stream can do, and more.

SMF record types that are critical for your company should use log streams that are backed by a staging data set so that every write to the structure is mirrored to DASD. That is, the log streams should be defined with STG\_DUPLEX(YES) and DUPLEXMODE(UNCOND).

If you have SMF records for which the loss of a small number of records in the unlikely event of a double failure is acceptable, the use of a CF-only log stream provides more throughput capacity than either of the other log stream options. As you can see in Appendix A, “Relative capacity measurements” on page 115, the throughput capacity of CF-only log streams is significantly better than that of either DASDONLY or CF log streams that use a staging data set.

Table 4-1 summarizes the capabilities of the types of log streams.

*Table 4-1 Determining the most appropriate type of log stream*

	<b>Supports sysplex-wide log stream?</b>	<b>Appropriate for critical SMF record types?</b>	<b>Can be accessed from multiple systems?</b>	<b>Supports extraction of data for just one system?</b>
DASDONLY	No	Yes	No	Yes
CF and staging data set	Yes	Yes	Yes	Yes
CF only	Yes	No	Yes	Yes

**Note:** To provide the optimum availability, every CF structure containing an SMF log stream should be used by more than one z/OS system, preferably residing on more than one processor. In such a configuration, if a system disconnects from a log stream unexpectedly, System Logger on one of the other connected systems automatically offloads the log stream to the offload data sets, protecting the log stream data in case a subsequent failure were to affect the CF.

The Sizing worksheet of the Stream\_Sizing spreadsheet contains a column called Log Stream Type (column G). When you decide which type of log stream to use for each of your SMF log streams, set these cells so that they accurately reflect the type of log stream that you will be using.

**Tip:** We found that it is important to carefully consider the names for your SMF log streams. They should be as short as possible to be easily remembered by the people trying to access the data in them, and the name should be such that it tells the user what data is contained in them. Also, the name should make it clear whether the log streams contain SMF data for just one system or from the entire sysplex. We found that it is a good idea to have a log stream naming convention similar to this:

*IFASMF.aaaaaaa.contents*

Where:

- ▶ IFASMF is a fixed value. The first qualifier of any SMF log stream must be IFASMF.
- ▶ aaaaaaaa is either the system name (for a log stream containing data from just one system) or the sysplex name. When you specify the log stream names in the SMFPRMxx member, you can use a system symbol (&SYSNAME. or &SYSPLEX.) as part of the log stream name.
- ▶ contents is an indicator of the type of SMF records that reside in that log stream.

For example, you might have two log streams:

- ▶ IFASMF.PRODPLEX.RMF, which contains RMF records for the entire sysplex
- ▶ IFASMF.PRODPLEX.DFLT, which contains all other SMF record types for the entire sysplex

The corresponding SMFPRMxx statements would look like:

- ▶ LSNAME(IFASMF.&SYSPLEX..RMF,TYPE(70:78))
- ▶ DEFAULTLSNAME(IFASMF.&SYSPLEX..DFLT)

### 4.2.3 Map out your SMF strategy

One of the objectives of your SMF logstream mode implementation plan should be to ensure that the cutover to logstream mode is as transparent as possible, with the minimum amount of possibilities for problems to occur.

To this end, we suggest that you start off by setting up definitions that will support both your initial move to logstream mode and your final target configuration.

For example, let us say that you have a two-system sysplex that you want to switch to logstream mode. Figure 4-1 depicts the current configuration.

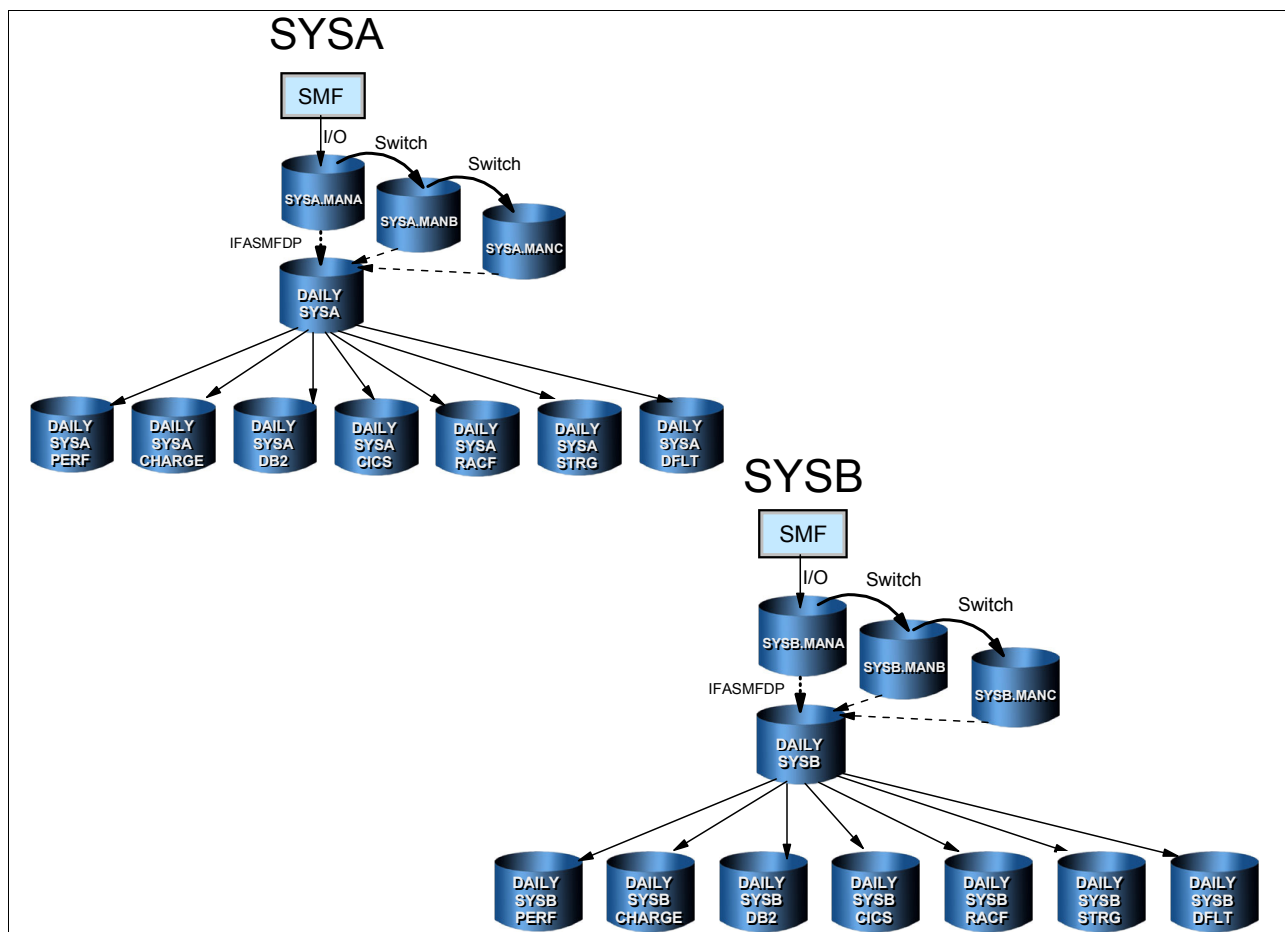


Figure 4-1 Current SMF configuration

You will see that both systems are writing to their respective SYS1.MAN data sets. When the data sets fill, they switch to a new one, and the contents of the full data set are immediately moved to a sequential data set (one per system). At the end of the day, the contents of those data sets are then split out across a number of GDGs, each containing a subset of the record types based on the needs of the users (for example, chargeback, storage management, DB2, performance, and so on). Generally, most of the processing of the SMF data only takes place after the SMF records reach these GDGs.

For the purposes of this example, let us say that you have decided that your ultimate target configuration is that the performance, DB2, and storage management SMF data will remain in log streams until they expire, while the chargeback, CICS, and RACF® records will continue to be moved to the existing GDGs. All the other miscellaneous SMF records will also continue to be written to a GDG. For the maximum flexibility, you also decide that all log streams will be CF log streams and will contain SMF records from every system in the sysplex.

Based on this, your target configuration would consist of seven CF log streams, and the log streams for performance, DB2, and storage management SMF records will be sized to hold six months' worth of SMF records. Remember that offload data sets are only allocated as they are needed, so if the log stream is sized to hold six months' worth of data, but the data is actually moved from the log stream every hour, you would simply end up with a lot fewer offload data sets.

Because you want to achieve the performance benefits of logstream mode as quickly as possible, while minimizing the impact on the users of your SMF records, you decide to move to your final configuration in two phases. The first phase is to implement approach 2 (simply replacing the SYS1.MAN data sets with log streams). The second phase is to move to approach 3 (where you start keeping some of the SMF data in log streams rather than moving it to the GDGs).

Based on your target configuration and your plan to implement approach 2 followed by approach 3, the first phase of your migration is to change SYSA so that it starts writing its SMF records to the seven log streams. Every hour, you run an IFASMF DL ARCHIVE job against each log stream. That ARCHIVE job would be set up so that it sends all records from all log streams to a sequential data set. A second step (DFSORT ICETOOL, for example) processes that data set and copies the SYSA records to the SYSA DAILY data set, and the SYSB records to the SYSB DAILY data set (even though there is no SYSB data being sent to the log stream yet) (Figure 4-2).

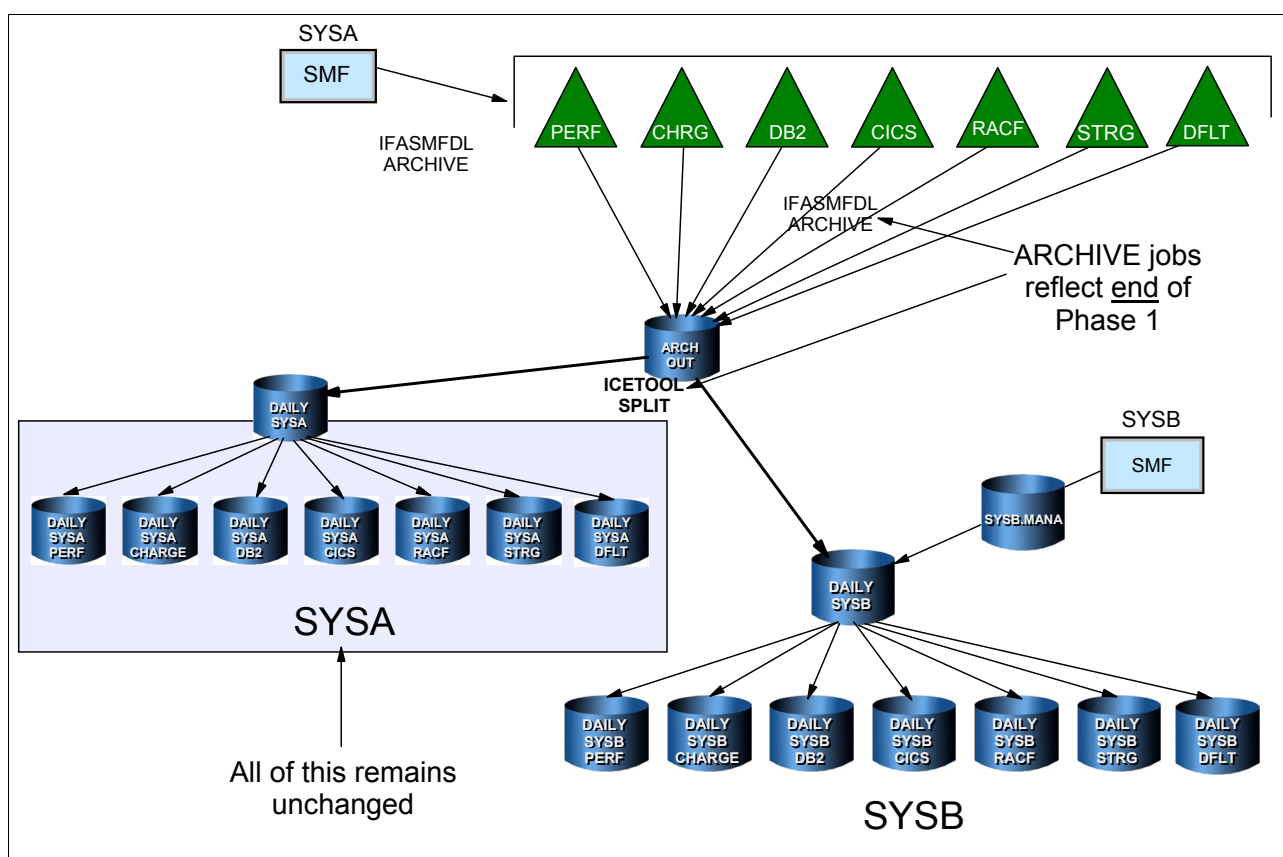


Figure 4-2 Cutover of SYSA to logstream mode

As you can see in Figure 4-2, the only real change here is that SYSA is now writing its SMF records to log streams instead of the SYS1.MAN data sets. However, all the jobs that run against the GDGs (or even the DAILY sequential data set) are completely unaffected by this change. If you are concerned that the ARCH OUT data set in Figure 4-2 would constitute a bottleneck, you could modify the jobs so that each log stream is archived to its own ARCH OUT data set, rather than sharing one between all log streams.

After you have run in this mode for some time, and the system programmers and operators have gained familiarity with logstream mode, you are then ready to switch the second system, SYSB, over to logstream mode (Figure 4-3).

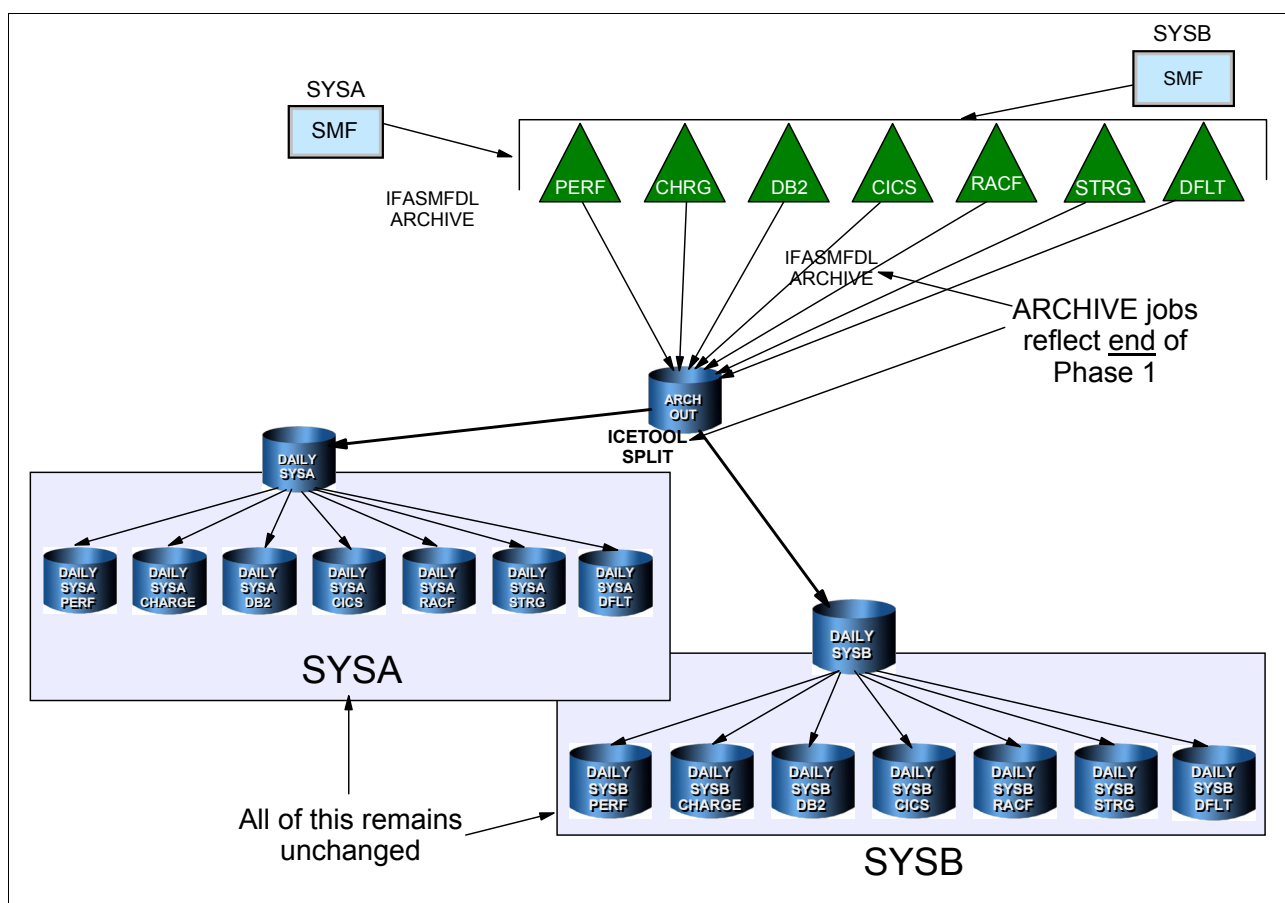


Figure 4-3 Cutover of SYSB to logstream mode

As you can see, the only thing that was changed here was that SYSB is now using the same SMFPRMxx member as SYSA so that it is also sending its SMF records to the same log streams. Everything else is unchanged. The jobs that were previously kicked off automatically when the SYS1.MAN data sets fill will no longer get started. Also, the IFASMF DL ARCHIVE and DFSORT ICETOOL jobs that you implemented when you moved SYSA to logstream mode do not require any changes because you set them up originally to handle data for both SYSA and SYSB. In addition, all the downstream jobs (the ones that use the SYSB SMF GDGs and the SYSB DAILY data set) are also unchanged.

You now have both systems writing to log streams, and all the SMF data subsequently getting moved to the same data sets that are used today. With minimal impact on all the users of your SMF data, you now have improved bandwidth for writing SMF data and hopefully have eliminated any problems that you might have been experiencing with SMF records being discarded because the SMF buffers are full.

However, you are not finished. If you recall, we said that the target configuration is approach 3, where the DB2, performance, and storage management SMF records will be kept in the log streams, while all the other SMF record types will continue to be moved to the existing GDGs.

To start the move to approach 3, you decide to cease archiving the performance SMF records from the log stream. There are two aspects to this change:

- First, you need to change your process so that the job that archives the performance SMF records out of the log stream will no longer be started.
- You also need to provide an alternative set of JCL that users of the performance SMF records can use to access the SMF records in the log stream rather than in the GDGs.

Figure 4-4 shows this change.

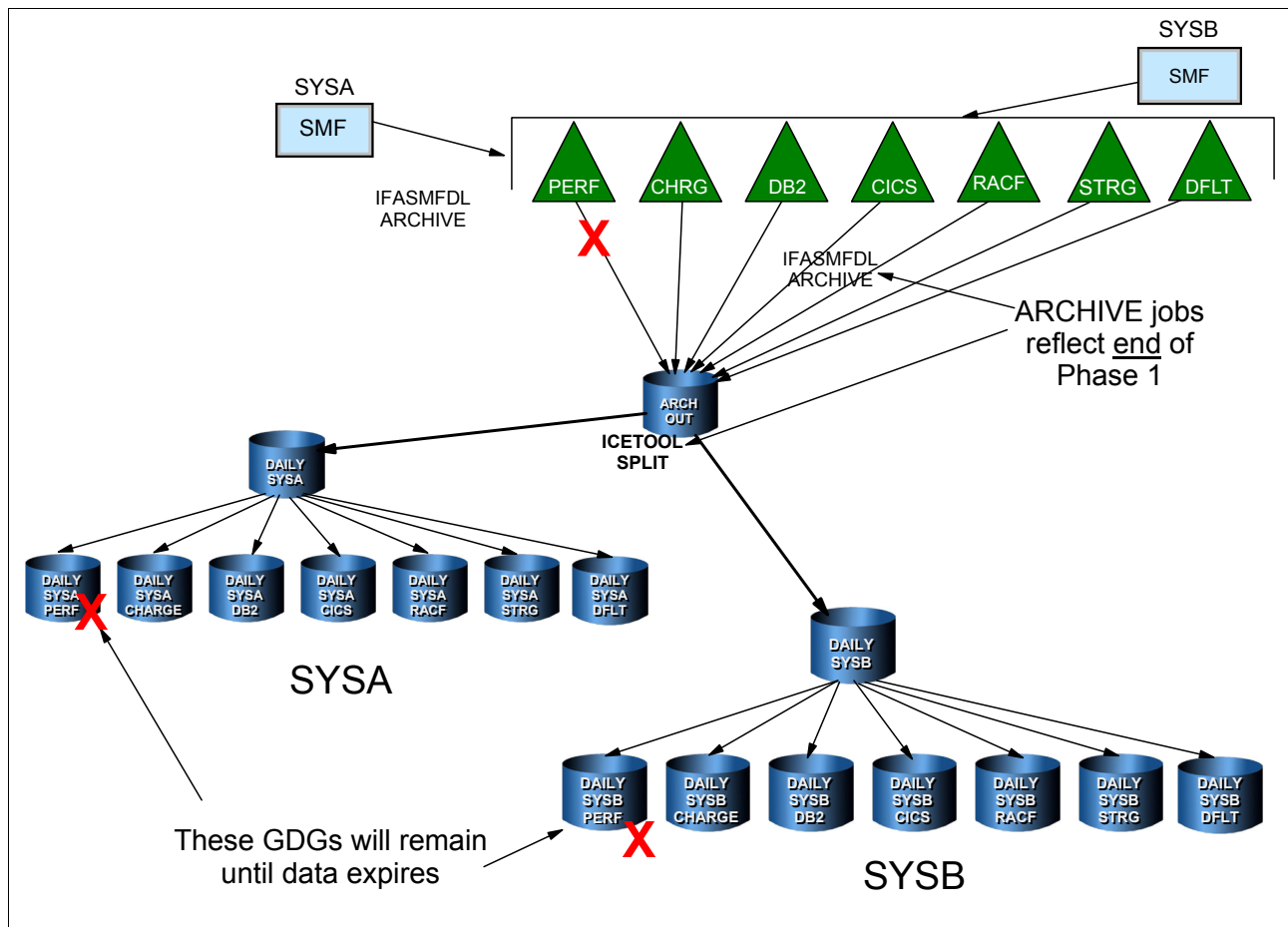


Figure 4-4 First step in move to approach 3

As you can see, stopping the movement of the performance SMF records from the log streams is as simple as stopping submission of the job that does the archive of the performance log stream.

After you are happy that everything is running OK, and your system programmers, operators, and SMF users are all happy with running in this mode, you can then extend this process to the other log streams that you want to use in this way.

We suggest that you draw a similar set of charts representing the steps in your migration process. When you have mapped out your target end point and are happy with the steps of how you will get there, you are now ready to continue with your cutover preparation.

## 4.2.4 Sizing the SMF log streams

There are three aspects to log streams that must be accurately sized to ensure optimum performance and availability:

- ▶ CF structure size (for CF log streams)
- ▶ Staging data set size (for DASDONLY and CF+staging log streams)
- ▶ Offload data sets (for all log streams)

To size the SMF log streams, you need to know:

- ▶ How long the records will be kept in the log stream
- ▶ The rate at which log blocks will be written to the log stream
- ▶ Whether the log stream will contain data from one system or multiple systems

For each log stream, you should know by now how you plan to manage it on a long-term basis. As discussed in 3.2.5, “Migration sequence” on page 44, we expect that most installations will initially continue to move all their SMF records to sequential data sets. However, we suggest that you size and define the log streams based on your longer term plans for each log stream. For example, if your longer term plan is to keep the performance SMF records in a log stream for six months, then size the performance log stream on that basis. The Manage\_SMF spreadsheet should contain information to help you identify how long you keep the record types that you will place in each log stream.

To help you size the log streams, we have provided a number of tools:

- ▶ A REXX exec called SMFRCSV that will convert an IFASMFDP report into CSV format. The IFASMFDP report that you use should represent a period of peak SMF record creation, and should ideally include SMF records from all the systems that will write to the associated log stream. Example 4-1 shows a sample job to create the IFASMFDP report.

*Example 4-1 Job to create IFASMFDP report that will be used for log stream sizing*

---

```
//SMFREP JOB (0,0),'IFASMFDP',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*
/* EXTRACT ONE HOURS WORTH OF SMF RECORDS FOR EVERY
/* SYSTEM IN THE SYSPLEX
/*
//STEP1 EXEC PGM=IFASMFDP
//SMFIN DD DISP=SHR,DSN=ZS9037.Z#@$.SMFDUMP(-1)
// DD DISP=SHR,DSN=ZS9037.Z#@2.SMFDUMP(-1)
// DD DISP=SHR,DSN=ZS9037.Z#@3.SMFDUMP(-1)
//SMFOUT DD DISP=(,PASS),SPACE=(CYL,(500,500),RLSE),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
        INDD(SMFIN,OPTIONS(DUMP))
        OUTDD(SMFOUT,TYPE(0:255))
        START(0900)
        END(1000)
/*
/* GET IFASMFDP REPORT FOR ONE HOURS WORTH OF SMF RECORDS FOR EVERY
/* SYSTEM IN THE SYSPLEX
/*
//STEP2 EXEC PGM=IFASMFDP
//SMFIN DD DISP=(OLD,DELETE),DSN=*.STEP1.SMFOUT
//SMFOUT DD DUMMY
//SYSPRINT DD DISP=(,CATLG),DSN=KYNEF.PRODPLEX.IFASMFDP,
// SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA,
```

```
//          RECFM=VB,LRECL=132,BLKSIZE=0
//SYSIN    DD      *
          INDD(SMFIN,OPTIONS(DUMP))
          OUTDD(SMFOUT,TYPE(0:255))
/*
```

---

The SMFRCSV REXX exec is described in “SMFRCSV program: Converting IFASMFDP output to CSV format” on page 141. Run that program against the IFASMFDP report data set and download the resulting CSV file to your PC.

- ▶ A spreadsheet called *Stream\_Sizing*. Using the CSV file created by the SMFRCSV program and the record type-to-log stream relationship that you defined in the *Manage\_SMF* spreadsheet as input, the *Stream\_Sizing* spreadsheet helps you identify the volume of SMF records being sent to each log stream.

This information helps you define the log streams and provides the information that you will use to size the CF structures.

At this time, follow the instructions in “*Stream\_Sizing* spreadsheet” on page 137 to complete the *Stream\_Sizing* spreadsheet. After you import the IFASMFDP CSV file into the spreadsheet and assign record types to log streams, you need to enter information in columns F to M of the Sizing worksheet. The values that you enter determine the results that are shown in columns O to T. You might find that fine-tuning is required (for example, increasing the *LS\_SIZE* value for a busy log stream should reduce the number of offload data sets that are required).

**Tip:** If the log streams will be used by multiple systems, either use an IFASMFDP report that represents all those systems, or adjust the scaling value in the L3 field on the Sizing worksheet. This is used to scale the results to allow for the other systems that will share the log streams.

If all your systems are similar (in terms of software mix), and the system that you input data from represents 1/10 of the total workload in the sysplex, then insert a value of 10 in that cell.

On the other hand, if the workload profile of each system is different, then it is easier and more accurate to simply include data from all systems in the IFASMFDP run that creates the IFASMFDP report that you import into the *Stream\_Sizing* spreadsheet.

## Sizing CF structures

The required size of a Logger structure is determined by the size of the blocks that are written to the log stream, the rate at which log blocks are written, and how long the blocks will be kept in the structure before being offloaded. Both the maximum block size and the write rate for each log stream are contained in the *Stream\_Sizing* spreadsheet. The CF Sizer tool assumes that SMF records will reside in the structure for 10 seconds. However, this value is built into the tool, so you do not need to worry about identifying the *correct* residency time.

We suggest that you place each SMF CF log stream in a separate CF structure. The storage in a System Logger structure is divided evenly between the connected log streams. If you have more than one log stream per structure, to optimize the use of storage in the structure, you need to use log streams that have similar write rates, and that is not always possible or easy to achieve. Also, placing just one log stream in each structure makes the structure sizing easier and does not have any disadvantages compared to having multiple log streams per structure.



Because some producers of SMF records create records in intermittent bursts, rather than a consistent flow, we also provide a REXX exec (called SMFRATE) that reads a data set containing SMF records and produces a report showing the write rate per hour and the write rate in the peak second for each record type encountered in that data set. The exec is documented in “SMFRATE program: Identifying SMF record volumes” on page 142.

To determine the structure sizes, take the Write Rate Blks/Sec (Column S) value from the Sizing worksheet in the Stream Sizing spreadsheet (or the high value from the SMFRATE spreadsheet, if that is significantly higher), together with the MAXBUFSIZE value (the suggested value is 65532) for each log stream, and input that to the CF Sizer tool, available on the web at:

<http://www-947.ibm.com/systems/support/z/cfsizer/>

The output from CF Sizer will contain the INITSIZE and SIZE values that you should use when defining the SMF log stream structures<sup>1</sup>. Run the tool now for each of your log streams, making a note of the INITSIZE and SIZE values for each structure.

### ***Creating the CFRM structure definition statements***

At this point, you should know how many and which log streams you will have and the mapping of log streams to CF structures. To make management of the SMF log stream environment as easy as possible, use structure names that clearly communicate the contents of the structure. A sample naming convention is:

*aaaaaa\_bbbbbb\_cccccc*

Where:

<b><i>aaaaaa</i></b>	Indicates the structure use (for example, SMF).
<b><i>bbbbbb</i></b>	Communicates the structure scope. Does it contain a log stream that only contains data from one system (in which case <i>bbbbbb</i> can be the system name) or from the entire sysplex (in which case <i>bbbbbb</i> can be the sysplex name)?
<b><i>cccccc</i></b>	Identifies the particular log stream that is contained in this structure. An easy way is to use the same “low-level qualifier” for the structure and the log stream that resides in that structure.

After you agree on a naming convention, you are ready to create the CFRM statements to define the structures in the CFRM policy. Note that you do not actually run the IXCMIAPU job until Chapter 5, “Implementation” on page 99.

You can use the following definition as a skeleton for defining your structures:

```
STRUCTURE NAME(SMF_scope_identifier)
  INITSIZE(xxxxx)          /* From CFSizer                */
  SIZE/yyyyy              /* From CFSizer                */
  PREFLIST(cf1,cf2)        /* Place your CF names here)   */
```

The defaults for all the remaining structure attributes should be acceptable. The ALLOWAUTOALT parameter should always be set to NO (the default) for System Logger CF structures.

<sup>1</sup> At the time of writing, CFSizer calculates a structure size that should result in an offload approximately every 10 seconds, assuming a HIGHOFFLOAD and LOWOFFLOAD thresholds of 80 and 0, respectively. This avoids overly large structures and minimizes the amount of SMF data that would be lost in the event of a double failure that affects the CF and a connected z/OS.

## Sizing offload data sets

Every log stream will have offload data sets. The offload data sets hold the log blocks between the time that they are moved from interim storage until they expire or are deleted. So, the size of the offload data sets is a function of the volume of data being written to the log stream and how long you plan to keep it in the log stream. Once again, the Stream\_Sizing spreadsheet will help. The columns of interest on the Sizing worksheet are:

- ▶ Number of days records will be kept in the log stream (column H)  
Specify the number of days that the SMF records will remain in the log stream before being archived or deleted.
- ▶ Number of days to keep offload data sets on primary DASD before migration (column M)  
Provide the target number of days that the offload data sets will remain on primary DASD before being migrated.
- ▶ Total size of log stream (column O)  
This is a calculated value. It includes interim storage, offload data sets on primary DASD, and migrated offload data sets.
- ▶ Primary DASD needed for offload data sets (column P)  
This is also a calculated value. This is the number of GB of DASD space required for the not-yet-migrated offload data sets.

This information helps you determine the amount of space that will be required for the offload data sets based on the attributes that you provide and the volume of data being written to each log stream.

SMF log streams are generally used in one of two ways:

- ▶ The data in the log stream is archived to sequential data sets and deleted from the log stream.
- ▶ The data is not moved to sequential data sets, and instead remains in the log stream until it expires.

In either case, the total amount of DASD space required to hold the SMF data should be roughly equivalent (the data will exist in either offload data sets or sequential data sets). Therefore, you might decide to place the offload data sets in the same SMS storage group that you use for your sequential data sets today, and possibly even use the same SMS management class.

**Important:** It is vital that the storage group used to contain the offload data sets does not fill, and that the SMF log streams do not run out of DSEXTENTS. If either of these events occur, Logger is unable to free space in interim storage, resulting in the SMF buffers associated with those log streams eventually filling. To give yourself as much time as possible to rectify this problem if it occurs, ensure that your automation product informs the appropriate people immediately if message IFA785E (SMF has used xx% of available buffer space) is generated.

For information about good sizes to use for the offload data sets, see the IBM Redbooks publication *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

## Sizing staging data sets

The other type of data sets that can be used by a log stream are the staging data sets. These are used by DASDONLY and CF log streams that use a staging data set. Size the staging data sets to be the same size as the structure associated with that log stream to ensure that

offloads are not triggered by the staging data set reaching the HIGHOFFLOAD threshold before the structure reaches that threshold. For CF log streams that use staging data sets, the IXCMIAPU utility will use the maximum structure size (as defined on the SIZE keyword on the structure definition in the CFRM policy) as the size when allocating the corresponding staging data sets if you do not specify a STG\_SIZE value. Another advantage of not specifying a STG\_SIZE value is that if the structure size is increased in the future, the next time that the staging data set is allocated, it automatically uses the new structure size value, so you do not need to worry about remembering to change the log stream definition if you change the structure size in the CFRM policy.

For a DASDONLY log stream, we suggest that you use the CF Sizer tool as if you were going to use a CF log stream, and use the SIZE value from that tool as the STG\_SIZE when defining that log stream in your LOGR policy, adjusting for the fact that the units on the STG\_SIZE keyword are 4 KB blocks, whereas the sizes returned by CFSizer are in terms of 1 KB blocks.

## Creating log stream definition statements

When you are happy with all the values in the Stream\_Sizing spreadsheet, you are ready to create the statements that will define your SMF log streams to System Logger. The syntax for the Logger definition statements is provided in the section titled “LOGR Keywords and Parameters for Administrative Data Utility” in *z/OS MVS Setting up a Sysplex*, SA22-7625. At this stage in the project, you are only creating the statements. Defining the log streams to System Logger takes place in Chapter 5, “Implementation” on page 99.

You first need to define the CF structures that will be used by any of the SMF log streams, using DEFINE STRUCTURE statements. The structure names need to match the structure names that you used in the CFRM policy. The following can be used as a skeleton for defining your structures to System Logger:

```
DEFINE STRUCTURE
    NAME(SMF_PRODPLEX_PERF)
    LOGSNUM(4) /* Suggest only 1 log stream per structure */
    MAXBUFSIZE(65532) /* IBM-recommended value */
```

Having created the statements to define the structures to Logger, you can now create the statements to define the log streams, using the following example as a skeleton:

```
DEFINE LOGSTREAM
    NAME(IFASMF.scope.lname) /* Insert log stream name here */
    STRUCTNAME(structure_name) /* Insert structure name here */
    LS_DATACLAS(LOGR24K) /* Use a data class with CISZ of 24KB */
    LS_SIZE(?????) /* From LS_SIZE column in Sizing worksheet */
    STG_DATACLAS(LOGR4K) /* Use a data class with CISZ of 4KB */
    STG_SIZE(?????) /* If this is a CF log stream, omit this */
    STG_DUPLEX(YES) /* Always specify YES for SMF log streams */
    DUPLEXMODE(COND) /* Specify UNCOND for critical log stream */
    DASDONLY(NO) /* Specify NO for CF log streams */
    HIGHOFFLOAD(60) /* Use value of 60 or less */
    LOWOFFLOAD(0) /* Specify small value here */
    RETPD(0) /* Use 0 if you will ARCHIVE records from this
               LS, otherwise specify how many days you plan
               to retain these record types in the log
               stream */
    AUTODELETE(NO) /* Only delete records when an ARCHIVE is run
                    against the log stream */
    HLQ(IXGLOGR) /* Perhaps use same HLQ as SMF seq DSETs */
```

There are certain considerations to keep in mind as you create the definition statements:

- ▶ Even though we suggest just one SMF log stream per CF structure, defining the structures to support more than one log stream (on the LOGSNUM parameter) gives you additional flexibility should you want to have more than one log stream per structure in the future.
- ▶ The log stream name *must* start with IFASMF.
- ▶ Specify an SMS data class for the offload data sets. That data class should have a CI Size of 24 KB and share options of (3 3).
- ▶ Specify an SMS data class for the staging data sets. That data class *must* have a CI Size of 4 KB and share options of (3 3).
- ▶ The offload data set size (LS\_SIZE) should reflect the value from the LS\_Size column of the Sizing worksheet of the Stream Sizing spreadsheet.
- ▶ For either type of CF log stream, omit the STG\_SIZE parameter and let the system use the current SIZE value from the corresponding structure. For a DASDONLY log stream, use a value that would give you the same size that the CF Sizer would provide for the SIZE keyword for the structure if this was a CF log stream.
- ▶ The RETPD should be 0 if your plan is to archive all the SMF records from that log stream. Specifying RETPD(0) means that the log blocks will be deleted whenever IFASMF DL issues a delete for them (either due to OPTIONS(DELETE) or OPTIONS(ARCHVE)). If you specify RETPD(5), no log blocks are deleted until they are five days old, even if you run an archive when they are just 30 minutes old.

When you are ready to keep the data in the log stream and stop using GDGs for this log stream, you need to update the log stream definition so that the RETPD reflects how long you will keep the records in the log stream.

- ▶ AUTODELETE should always be set to NO for a log stream that you will be running IFASMF DL ARCHIVE or DELETE against. When you switch to keeping data in the log stream, you need to change this to YES.

**Important:** The RETPD and AUTODELETE parameters must be used together.

If your plan is to move the SMF records out of the log stream to a sequential data set (so that IFASMF DL will be responsible for deleting the records), turn off Logger-driven deletion by specifying RETPD(0) and AUTODELETE(NO).

On the other hand, if your strategy is to leave the data in the log stream until it expires, set the appropriate RETPD(xx) value (to tell Logger at which point it should delete those log blocks) and specify AUTODELETE(YES) to turn on Logger-driven deletion for that log stream.

If you have a mismatch of AUTODELETE and RETPD parameters, you might find that log blocks are being retained longer than you had planned, or are being deleted sooner than you expected. For example, let us say that you specify RETPD(0) and AUTODELETE(YES) and your intention is to run hourly archives to move the records to sequential data sets, however, for some reason, the archives do not work. Because you specified AUTODELETE(YES), the log blocks are immediately eligible for deletion by System Logger, even though they have not yet been saved to the sequential data sets.

- ▶ The HIGHOFFLOAD and LOWOFFLOAD parameters need to reflect how the log stream is being used.

For a CF log stream, the entries and elements in the CF structure are freed up as each block is physically deleted or moved to DASD during System Logger offload processing.

However, for a DASDONLY log stream, the space that is freed up by the offload process is not available for reuse until the entire offload has completed.

SMF log streams are typically *funnel*-type log streams. The data written by SMF to the log stream is rarely read back again and is deleted some time (hours or even months) in the future. Therefore, the aim for these types of log streams is that the data written to interim storage is moved to the offload data sets before interim storage is filled. The System Logger offload starts when the HIGHOFFLOAD threshold is reached and ends when LOWOFFLOAD is reached. We have to ensure that the offload is completed before interim storage reaches 100% full.

In Figure 4-5, a DASDONLY log stream with a 120 MB staging data set is being written to at a rate of 2 MBps (that is, it takes 60 seconds to fill the data set). With a HIGHOFFLOAD value of 80% and a LOWOFFLOAD value of 20%, there are only 12 seconds for the offload process to reach the LOWOFFLOAD point and make the control intervals available in the staging data sets. By setting a lower value for both thresholds, you can increase the time available for the offload.

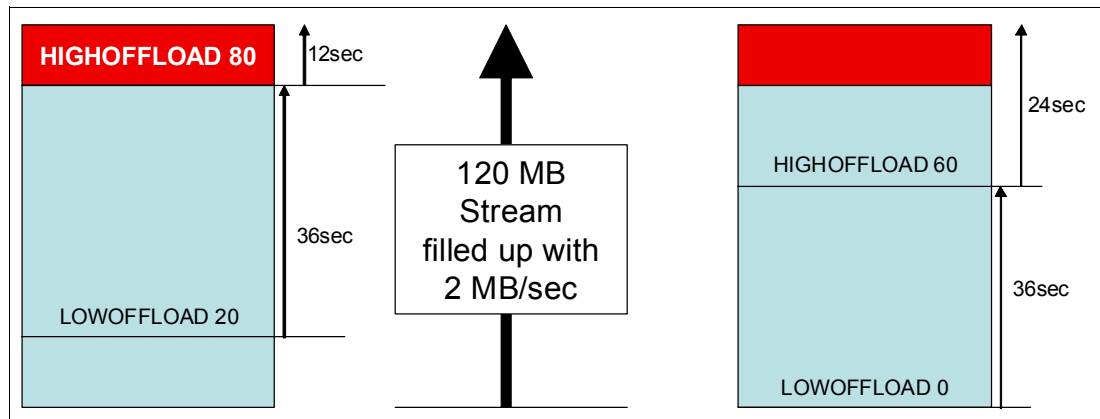


Figure 4-5 Choose the right offload values

Because the data in a funnel-type log stream is rarely accessed again within a few seconds of being written, there is no benefit in attempting to keep some of the SMF data in interim storage. Therefore, we suggest a HIGHOFFLOAD value not higher than 60 and a LOWOFFLOAD value of 0.

- The HLQ parameter requires consideration.

Because the Logger offload data sets are Virtual Storage Access Method (VSAM) data sets, they cannot be accessed unless the catalog that they are cataloged in is available. If that catalog were to be corrupted, you would need to recover it. However, certain catalog recovery tools use SMF records to perform the recovery. So you could end up in a situation in which you need the SMF records to recover the catalog, but you cannot access the SMF records because the catalog that points to the data sets containing those SMF records is broken.

For this reason, we suggest extracting the SMF types 60, 61, 65, and 66 to *two* sequential data sets (each with a different HLQ and cataloged in different user catalogs), and that neither of those data sets use the same HLQ or the same user catalog as the offload data sets.

A detailed description of how to recover ICF Catalogs can be found in *ICF Catalog Backup and Recovery: A Practical Guide*, SG24-5644, and *ICF Catalog Backup and Recovery: Catalog RecoveryPlus Update*, REDP-4212.

► **MAXBUFSIZE** (DASD-only log streams)

The MAXBUFSIZE value should be specified for DASDONLY log streams. This defines the maximum log block size, in bytes, that the system can write to this log stream. IBM suggests a MAXBUFSIZE of 65532, but it must be at least 33024. For a CF log stream, the MAXBUFSIZE that is specified on the DEFINE STRUCTURE statement will be used for all log streams in that structure, so it should not be specified on the DEFINE LOGSTREAM statement.

There is one additional attribute that you might consider, and that is OFFLOADRECALL. This specifies what Logger should do when it attempts to do an offload and it finds that the target offload data set has been migrated. Specifying OFFLOADRECALL(NO) results in Logger allocating a new offload data set rather than waiting for the original target data set to be recalled. This can be effective in reducing the chance of an offload being delayed because of long recall times. Realistically, given the volume of data that is likely to be written to each SMF log stream, it is unlikely that there will be a long enough interval between offloads to give a data set a chance to be migrated before it fills and Logger switches to a new one. Nevertheless, the use of OFFLOADRECALL(NO) might provide a small amount of extra insurance against delays during an offload.

## 4.2.5 Preparing SMFPRMxx members

There are two likely ways to prepare your SMFPRMxx members for the cutover to SMF logstream mode. One is to use a single SMFPRMxx member that contains definitions for both log streams and data sets. You would then use the **SETSMF RECORDING(xxx)** command to switch recording modes. The other way is to have two members, with one member containing a RECORDING(LOGSTREAM) statement, and the other member containing RECORDING(DATASET).

The obvious advantage of using a single member and then switching with the SETSMF command is that you do not have the overhead and risk of trying to keep two members in synch with each other. However, the downside of this approach is that the SETSMF command will currently only be allowed if you specify PROMPT(xxx) in the SMFPRMxx member, and this results in an operator prompt every time that you IPL and every time that you switch SMFPRM members using the SET SMF=XX command. If you are using z/OS 1.12, you can use the AUTO REPLY capability to automatically respond to these prompts. But prior to that release, the operators need to respond manually, as your automation product is not available that early in the IPL process. While this is not ideal, you should be able to revert to NOPROMPT once all your systems have migrated to logstream mode, and you do not need to revert to dataset mode.

Despite this, if you decide to use just a single SMFPRMxx member, then it is a good idea to define the RECORDING value as a system symbol in your IEASYMxx member. This allows you to use a single SMFPRMxx member for all systems in the sysplex and have the RECORDING value resolve to LOGSTREAM on the systems that are in log stream mode, and DATASET on the remaining members of the sysplex. Example 4-2 shows an extract from a sample SMFPRMxx member. If you use this mechanism in your SMFPRMxx member, remember to update the IEASYMxx member every time that you switch recording modes, to ensure that the system comes back up in the same mode in case of an unexpected IPL.

*Example 4-2 Sample SMFPRMxx statements*

---

```

ACTIVE                                /* ACTIVE SMF RECORDING                */
    SID(&SYSNAME(1:4))                /* SYSTEM ID IS &SYSNAME                */
    RECORDING(&SMFMODE.)              /* SMF LOGR RECORDING ACTIVE            */
    DSNAME(SYS1.&SYSNAME..MANC, /* SMF DATA SET NAMES                */
           SYS1.&SYSNAME..MAND) /* SMF DATA SET NAMES                */
    LSNAME(IFASMF.&SYSPLEX..PERF,TYPE(16,50,70:79,99,113))
    LSNAME(IFASMF.&SYSPLEX..DB2,TYPE(100:102))
    LSNAME(IFASMF.&SYSPLEX..CICS,TYPE(110))
    LSNAME(IFASMF.&SYSPLEX..STRG,TYPE(42))
    LSNAME(IFASMF.&SYSPLEX..CHRG,TYPE(30),BUFUSEWARN(10))
    LSNAME(IFASMF.&SYSPLEX..SEC,TYPE(80:83),NOBUFFS(HALT))
    DEFAULTLSNAME(IFASMF.&SYSPLEX..DFLT)
    PROMPT(ALL)                       /* NO PROMPT                            */
    NOBUFFS(MSG)
    BUFUSEWARN(50)

```

---

To prepare the SMFPRMxx member to support both dataset and logstream modes, add LSNAME and DEFAULTLSNAME statements *in addition to* the existing DATASET statements. Create an LSNAME statement for each log stream that is defined in your Stream\_Sizing spreadsheet, remembering to include a DEFAULTLSNAME statement that defines the log stream that will hold all the other SMF record types.

**Note:** For SMF logstream mode, you can direct record types to particular log streams using the TYPE subparameter on LSNAME. You still select the records that you want to write with the TYPE/NOTYPE option of SYS or SUBSYS. This means that it is possible to specify record types on the TYPE subparameter of LSNAME that the system is not actually recording, because they are not specified on SYS or SUBSYS.

If you have SMF record types that are deemed to be really critical to your enterprise, you might consider sending those record types to *two* log streams, with each log stream using a CF structure in separate CFs, and duplexing to staging data sets in two storage control units.

Also review the values on your NOBUFFS, BUFUSEWARN, and MAXDORM keywords at this time.

**Important:** You cannot specify subtypes on the TYPE parameter for LSNAME or DEFAULTLSNAME.

## **NOBUFFS**

Starting in z/OS 1.12, you have the ability to specify different NOBUFFS values for each log stream. Specify a default value of NOBUFFS(MSG), and then override that by specifying NOBUFFS(HALT) on the LSNAME statement for any log stream that contains critical SMF record types.

## **BUFUSEWARN**

Also new in z/OS 1.12 is the ability to specify a BUFUSEWARN value at the log stream level. To allow yourself ample time to investigate any potential problems, it is a good idea to specify a relatively small value (for example, 20 or less) so that if the buffers start filling up (indicating a potential offload problem) you have a chance to address the problem before the buffers fill. Ensure that you add the BUFUSEWARN message (IFA785E) to your automation package to ensure that if any warning messages *are* issued that you will be made aware of them<sup>2</sup>.

To allow for sharing SMFPRMxx members between V1R12 and V1R10/V1R11 systems, apply APAR OA30848 to the pre-release 1.12 systems. This APAR ignores the new SMFPRMxx options when they are read by back-level systems.

## **MAXDORM**

When running in dataset mode, you normally want to ensure that SMF always fills its buffers before writing to the SYS1.MAN data sets. This is to ensure that the SMF I/Os are as efficient as possible. To achieve this, the MAXDORM keyword is often set to a high value (MAXDORM controls how long SMF waits before closing a buffer and writing it).

However, when running in logstream mode, the need to write full buffers is not as great. In fact, because certain log streams might have relatively low write rates (because they only contain a subset of the records that are being written to your SYS1.MAN data sets today), you might want to use MAXDORM to ensure that the SMF records are externalized in a timely manner. Additionally, there are advantages to ensuring that each log block does not contain SMF records from a large span of time (this was discussed in 1.4, “SMF support for log streams” on page 9).

For these reasons, you might want to lower your MAXDORM value to one minute or less by specifying MAXDORM(0100) in SMFPRMxx.

When SMF logstream mode was originally delivered in z/OS 1.9, MAXDORM was not supported for log streams. However, this support was subsequently rolled back to that release with APAR OA27037.

**Note:** MAXDORM is specified as mmss, and the default value is 3000 (30 minutes). We suggest that a much smaller MAXDORM value is used when SMF is running in logstream mode.

## **Define IFASMFDL/IFASMFDL dump exits**

During your testing prior to implementing logstream mode in production, you are likely to have a need to understand which SMF records are contained in a data set or log stream that is input to the IFASMFDL and IFASMFDL programs, as well as the contents of the data sets created by those programs (just to ensure that they are working as you expect). Appendix E, “Additional material” on page 135, contains three sample user exits that work with both IFASMFDL and IFASMFDL. The exits are described in “Sample IFASMFDL exits” on page 147. For the exits to work, they must be defined in the SMFPRMxx member, using the SMFDPEXIT and SMFDLEXIT statements (Example 4-3).

### *Example 4-3 Defining IFASMFDL and IFASMFDL exits in SMFPRMxx*

```
SMFDLEXIT(USER1(SMFDPUX1))
SMFDLEXIT(USER2(SMFDPUX2))
SMFDLEXIT(USER3(SMFDPUX3))
```

<sup>2</sup> Prior to z/OS 1.12 SMF, the BUFUSEWARN function did not apply to log streams, so no message would be issued prior to the buffers filling when in logstream mode. However, you *can* issue a D SMF command to determine how much space in each log stream’s buffer is currently in use.



```
SMFDPEXIT(USER1(SMFDPUX1))  
SMFDPEXIT(USER2(SMFDPUX2))  
SMFDPEXIT(USER3(SMFDPUX3))
```

---

Note that if you have your own IFASMFDP exits that you are using in dataset mode and want to continue to use them when you move to log stream mode, it is possible to use the same exits without making any changes. However, if you want the exit to use the log stream name (rather than the DDNAME), you need to make a small change to the exit program.

## SMF exits

No changes are required to any SMF exits to get them to work in logstream mode. Any exit that works in dataset mode continues to work in logstream mode.

However, there is a special consideration for the IEFU29 exit. This exit is invoked any time that a SYS1.MAN data set fills. Because you will not be using these data sets when in logstream mode, this exit will no longer be invoked. Because IEFU29 will no longer be invoked when you are running in logstream mode, SMF provides an equivalent exit, called IEFU29L for logstream mode. The IEFU29L exit is invoked any time that an **I SMF** command is issued.

You need to review the function of your existing IEFU29 exit. If that exit simply submits a job to empty the SYS1.MAN data set, then you might replace that function by having your batch scheduling product submit IFASMFDP ARCHIVE jobs on a regular basis to archive any log streams that you want to move to sequential data sets.

Alternatively, you can have your automation product issue an **I SMF** command on a regular basis and have the IEFU29L exit carry out whatever functions were provided by your IEFU29 exit.

While a sample IEFU29L exit is not currently delivered in SYS1.SAMPLIB, one is provided in Appendix E, “Additional material” on page 135. For more information, see “Implementing the sample IEFU29L exit” on page 145 and *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

## 4.2.6 Review configuration

Because the use of SMF logstream mode is a significant change to the system, and because it might result in significantly more intensive use of System Logger, it is prudent to check certain aspects of your system setup before you proceed with the cutover to SMF logstream mode.

### Planning for processor storage requirements

SMF uses buffers to help it handle times when SMF records are being passed to it faster than they can be externalized (either to the SYS1.MAN data sets or to System Logger). In normal operation, and assuming that wherever the SMF records are being stored can keep up, not many buffers will be used. However, if the arrival rate is consistently higher than the rate at which they can be externalized, *or* if there is a delay in externalizing the records, then the number of buffers that are in use can increase dramatically.

In dataset mode, SMF has a maximum of 1 GB of buffer space, and those buffers are backed by pageable storage. In logstream mode, SMF uses data spaces to hold the buffers for SMF log blocks that have not been passed to System Logger yet. Each log stream has its own 2 GB data space, and the storage used for those buffers is Disabled Reference (DREF) storage, which is *not* pageable. While it is possible to control the maximum amount of storage

used for SMF buffers when running in dataset mode (using the BUFSIZMAX keyword in SMFPRMxx), at the time of writing it is not possible to specify any value less than 2 GB per log stream when in logstream mode.

At initialization, SMF obtains enough storage for 100 buffers for each log stream. The size of each buffer is determined by the MAXBUFSIZE that is used when the log stream is defined to System Logger, but we assume that you will use the suggested value of 64 KB. This means that SMF will obtain 6.4 MB of non-pageable storage for each log stream, so each log stream will require a minimum of 6.4 MB of real storage.

For normal activity, SMF is not likely to require any more than this. However, if you are generating SMF records at high volumes, or if there is a delay in the offload process from interim storage, it is possible for SMF to obtain up to 2 GB of central storage for buffers *per log stream*.

Also consider the fact that the more log streams that you have in place, the more DREF storage SMF consumes due to the fact that each log stream has its own data space, each with a minimum of 100 buffers per log stream. This can potentially change to gigabytes per log stream if Logger experiences problems in offloading from interim storage. You need to balance the flexibility and performance that the use of many log streams can provide against the potential real storage cost of a large number of SMF log streams.

For these reasons, be wary of implementing logstream mode in a system that is already storage-constrained. Additionally, for any system, to enable you to address any storage shortages as quickly as possible, it is prudent to ensure that any Logger or Real Storage Manager (RSM) messages<sup>3</sup> that indicate a problem with offload processing or storage shortages be defined to your automation and that they are brought to the attention of the appropriate personnel immediately.

## **Review System Logger setup**

You probably have your LOGR Couple Data Sets already defined if you have System Logger functions active on your system. Prior to implementing SMF logstream mode, it is wise to review your existing Logger setup.

---

<sup>3</sup> RSM messages generally start with IRA (for example, IRA100E SQA Shortage).

To start, you want to display how your current LOGR CDSs are defined. To do this, issue a D XCF,C,TYPE=LOGR command (Figure 4-6).

```
D XCF,C,TYPE=LOGR
IXC358I 11.37.27 DISPLAY XCF 507
LOGR COUPLE DATA SETS
PRIMARY   DSN: SYS1.XCF.MAXSYS12.LOGR01
          VOLSER: #@X1      DEVN: D20F
          FORMAT TOD      MAXSYSTEM
          05/27/2008 09:18:07 12
          ADDITIONAL INFORMATION:
            LOGR COUPLE DATA SET FORMAT LEVEL: HBB7705
            LSR(1000) LSTRR(600) DSEXTENT(50)
            SMDUPLEX(1)
ALTERNATE DSN: SYS1.XCF.MAXSYS12.LOGR02
          VOLSER: #@X2      DEVN: D30F
          FORMAT TOD      MAXSYSTEM
          05/27/2008 09:18:09 12
          ADDITIONAL INFORMATION:
            LOGR COUPLE DATA SET FORMAT LEVEL: HBB7705
            LSR(1000) LSTRR(600) DSEXTENT(50)
            SMDUPLEX(1)
LOGR IN USE BY ALL SYSTEMS
```

Figure 4-6 Displaying LOGR CDS attributes

This provides information about how the current CDSs are defined (in terms of the number of structures, log streams, and DSEXTENTs that they are formatted to support) and where they are allocated. It is important that there is no single point of failure between the two CDSs, and that the CDSs are formatted with the SMDUPLEX(1) keyword (even if you are not using System Managed Duplexing).

The one other piece of information that you need before proceeding is to determine the current utilization of the CDSs. To do this, run an IXCMIAPU list job (Example 4-4).

Example 4-4 Job to list LOGR CDS utilization

```
//KYNEFI JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
```

The resulting report contains detailed information about all your defined log streams (which we are not interested in at the moment), but near the top of the report there is a summary of the CDS usage (Figure 4-7).

LOGR Inventory Record Summary:		
LOGR COUPLE DATA SET FORMAT LEVEL: HBB7705		
/* Functional Items: */		
/* SMDUPLEX(1) */		
Type	Formatted	In-use
-----	-----	-----
LSR (Log Stream)	1,000	889
LSTRR (Structure)	600	275
DSEXTENT (Data Set Extent)	50	8

Figure 4-7 Displaying LOGR CDS utilization

Ensure that there is a sufficient difference between the formatted and in-use values to allow you to define the structures and log streams that you will be using for SMF log streams.

These are the keywords to consider:

<b>LSR</b>	This parameter specifies the maximum number of log streams that can be defined in the LOGR policy.
<b>LSTRR</b>	The LSTRR parameter defines the maximum number of structures that can be defined to the LOGR policy.
<b>DSEXTENT</b>	<p>When a log stream needs more than 168 offload data sets, it needs additional directory extents to maintain the inventory for the next 168 data sets. These extents are given from a common pool and assigned to a particular log stream. A DSEXTENT will not be shared between log streams once they are in use. When all the data sets in a directory extent are physically deleted, the directory record is returned to the common pool.</p> <p>DSEXTENT is the total number of available data set extents for all log streams defined in this couple data set.</p> <p>Depending on the size and the number of offload data sets that you expect for the log stream, this value might need to be adjusted.</p>

SMF logstream mode implementation might result a significant number of log stream offload data sets. The Stream\_Sizing spreadsheet will help you identify how many log streams and structures you will be defining. The LSR (Max logstream number) or LSTRR (Max Structures) values might need to be adjusted to a higher value that would cater for your existing log stream configuration, plus the additional SMF log streams that you will be adding.

The Stream\_Sizing spreadsheet also shows you the number of DSEXTENTs that will be required for each log stream based on the volume of data being sent to that log stream and

your definition of how long the data will remain in the log stream (this value is contained in Column T for each log stream on the Sizing worksheet).

If the difference between the formatted and the in-use log streams is not enough to accommodate the DSEXTENTS values in the Stream\_Sizing spreadsheet, you will need to format a new set of LOGR CDSs with appropriate DSEXTENT values. Example 4-5 shows a sample job to define a new LOGR CDS.

*Example 4-5 Define new LOGR CDS including a spare*

---

```
//STEP1      EXEC PGM=IXCL1DSU
//SYSPRINT DD  SYSOUT=*
//SYSIN       DD  *
      DEFINEDS SYSPLEX(@$#PLEX)
              MAXSYSTEM(4)
              DSN(SYS1.XCF.LOGRPRI) VOLSER(@$#X1)
              CATALOG
      DATA TYPE(LOGR)
              ITEM NAME(LSR) NUMBER(1200)
              ITEM NAME(LSTRR) NUMBER(120)
              ITEM NAME(DSEXTENT) NUMBER(50)
              ITEM NAME(SMDUPLEX) NUMBER(1)
      DEFINEDS SYSPLEX(@$#PLEX)
              MAXSYSTEM(4)
              DSN(SYS1.XCF.LOGRALT) VOLSER(@$#X2)
              CATALOG
      DATA TYPE(LOGR)
              ITEM NAME(LSR) NUMBER(1200)
              ITEM NAME(LSTRR) NUMBER(120)
              ITEM NAME(DSEXTENT) NUMBER(50)
              ITEM NAME(SMDUPLEX) NUMBER(1)
      DEFINEDS SYSPLEX(@$#PLEX)
              MAXSYSTEM(4)
              DSN(SYS1.XCF.LOGRSPR) VOLSER(@$#X3)
              CATALOG
      DATA TYPE(LOGR)
              ITEM NAME(LSR) NUMBER(1200)
              ITEM NAME(LSTRR) NUMBER(120)
              ITEM NAME(DSEXTENT) NUMBER(50)
              ITEM NAME(SMDUPLEX) NUMBER(1)

/*
```

---

Create the JCL now to create the new CDSs. You will use that JCL in Chapter 5, “Implementation” on page 99, as you prepare for the cutover to logstream mode.

For more detailed information, see *z/OS MVS Setting up a Sysplex*, SA22-7625; *z/OS MVS System Commands*, SA22-7627; and the IBM Redbooks publication *System z Parallel Sysplex Best Practices*, SG24-7817.

See 4.2.8, “Automation considerations” on page 75, for automation considerations related to messages that warn you of impending shortages of DSEXTENTS in the LOGR CDS.

## Changes to active log streams

With good planning and design, your log streams should be set up correctly from the beginning. While it is possible (and sometimes necessary) to subsequently change the

attributes of a log stream, switching a log stream between DASDONLY and CF-based can be disruptive and therefore should be avoided if possible.

Many log stream attributes can be updated dynamically using the IXCMIAPU utility. However, depending on the attribute being changed, a subsequent action is usually required to actually make the change active. For example, attributes related to offload data sets are typically activated when the next offload data set for that log stream is allocated. For CF structure-based log streams, changes to many attributes can be activated by performing a structure rebuild. For detailed information about which attributes can be changed dynamically and what is required to activate the change, see the sections titled “Upgrading an existing log stream configuration” and “Updating a log stream’s attributes” in *z/OS MVS Setting up a Sysplex*, SA22-7625.

There are certain log stream attributes that require that *all* connectors disconnect from the log stream to activate the change. At the time of writing, those attributes are DESCRIPTOR, GROUP, and STRUCTNAME. For the SMF log streams, the only way to get SMF to disconnect from a log stream is to stop using that log stream. How disruptive such a change is depends on whether you keep the SMF data in that log stream only briefly (before archiving to a sequential data set) or if you keep the data there on a permanent basis.

If you regularly archive the data from the log stream, you can briefly change SMF to write to a temporary log stream, empty the one you are changing (using IFASMF DL ARCHIVE), change the attributes, switch SMF back to the original log stream, and empty the temporary log stream. A sample process is described in Appendix D, “Changing log stream attributes” on page 133.

However, if you keep SMF data in the log stream until it expires, making a change that requires disconnection from the log stream is more complex. The only way to disconnect SMF from a log stream is to direct those SMF records to a different log stream or to switch back to dataset mode. Remember that if this is a sysplex-wide log stream, then SMF on every system must disconnect from the log stream. You now have the challenge that SMF records from the period while you are making the change are being written elsewhere, and realistically, there is no way to get them back into the *normal* log stream. So, until those SMF records expire, the records created during the window when you are making the change will need to be retrieved from a different location than all the other records of the same type. This is not a desirable situation, so try to avoid having to change any of these attributes if at all possible.

## Review SMS setup

When you define a log stream, you have the option to specify a particular SMS data class, management class, and storage classes. The data class is particularly important because this is the only way to ensure that the correct VSAM SHROPTIONS is used for Logger offload and staging data sets. All log stream definitions should refer to a data class that is defined with SHROPTIONS of (3 3).

The data class should also be used to specify the CI size for the data sets. The default value is 4 KB, which is appropriate for the staging data sets (in fact, 4 KB is the *only* supported CI size for staging data sets). However, to provide the best possible performance, the offload data sets should have a CI size of 24 KB. So this means that you really need at least two data classes:

- ▶ One that specifies a CI Size of 4 KB
- ▶ One that specifies 24 KB

Additionally, if the data class does not contain any size value, and no LS\_SIZE is specified when the log stream is defined, a default value of just two tracks will be used, and this is not nearly sufficient for even the smallest SMF log stream. We suggest that the data class used

by offload data sets specify a value that matches the LS\_SIZE value (column J) in the Stream\_Sizing spreadsheet. If you assign different LS\_SIZE values to each log stream in the spreadsheet, set up the data class to use a size value that is at least as large as the smallest of the LS\_SIZE values in your spreadsheet, and then specify the appropriate LS\_SIZE value when you define each log stream.

Because the best way to specify the size of a staging data set associated with a CF log stream is to let it default to the structure size, the data class that you specify on the STG\_DATACLAS keyword should not contain any size value. However, this means that you need to remember to specify a STG\_SIZE value for any DASDONLY log streams.

If you assign an SMS management class to the offload data sets (using LS\_MGMTCLAS), ensure that the attributes of the management class (in terms of how long the data set should remain in primary DASD before being migrated) match the values that you used in the Stream\_Sizing spreadsheet. If there is a mismatch, you might find that you require more space on the primary DASD than you expected. If a user runs a report against SMF data in log streams, this can potentially result in a large number of recalls. There is no possibility to limit these recalls, so calculate the storage available for the offload data sets generously.

Consider specifying a larger-than-normal number of days-on-primary in the following cases:

- ▶ If the SMARTENDPOINT capability is not available on your system.
- ▶ If you are not comfortable that the users that run jobs against *old* SMF data will always remember to specify SMARTENDPOINT.
- ▶ If you leave data in a log stream for a long time (so that there are many migrated offload data sets in that log stream) and then do an IFASMF DL ARCHIVE (because the ARCHIVE will read all the way to the end of the log stream, even if you only request to ARCHIVE one day's worth of data.).

We do *not* recommend that you do this. If you are going to archive data from the log stream, we believe that it is better to do so shortly after it is written to the log stream. If you are going to leave data in the log stream for a long time, you need to consider what is the point of archiving it to a sequential data set at that point, instead of just leaving it in the log stream until it expires.

In any of these situations, processing old data in log streams can result in significant numbers of HSM recalls.

Another item related to migration and recall to consider is that when HSM recalls an offload data set, media manager turns on the change bit for this data set. This can result in an increase in your HSM backup activity because the data set now needs to be backed up again before data can be migrated.

The problem can be bypassed by assigning a different management class in the management class ACS routine for the logger offload data sets, when these data sets are recalled from HSM. In this management class, backup is not required, and PRIMARY DAYS NON-USAGE should be set to 0.

The use of extended format striped data sets for staging and offload data sets can help to improve the performance of busy log streams. Striping spreads a data set over a specified number of volumes so that I/O parallelism can be exploited. The SMS Data Set Name Type attribute in the SMS data class controls whether a data set can be extended format. The number of stripes that will be allocated to the data set is based on the Sustained Data Rate attribute in the SMS Storage Class.

### Ensure DASD pool for staging and offload data sets is large enough

It is vital that the storage group used for the offload data sets never fills up. If Logger is unable to allocate a new offload data set, it will be unable to offload log blocks from interim storage. This eventually results in the SMF buffers filling. Even if it does not get to the point that the buffers fill, remember that the SMF buffers reside in non-pageable DREF storage, so having large numbers of buffers can cause your system to become storage-constrained.

Additionally, when determining the amount of primary DASD space that you will require for the staging and offload data sets, remember to allow for offload data sets that were migrated (because they contain old data), but then were subsequently recalled because they contained SMF records that are required by a report program.

### Ensure user catalog is large enough

Depending on how long you plan to keep the SMF data in the various log streams, you might potentially have a large number of offload data sets when you reach the end of your logstream mode migration project. Therefore, check the user catalog that the offload data sets will be cataloged in to ensure that it has sufficient free space and room for growth.

## 4.2.7 Operator education

Your operators are likely to be familiar with how SMF works (at least in relation to managing the SYS1.MAN data sets). However, they might be less familiar with how System Logger works, and they probably have no experience with SMF in logstream mode.

Prior to migrating to logstream mode, certainly for your production system, provide education for the operators who will cover the following topics:

- ▶ What SMF data is, what SMF records are used in your enterprise, why SMF data is so important to you, and an overview of the current SMF processing (what happens when the SYS1.MAN data sets fill, how those processes get triggered, and the life cycle of SMF data).
- ▶ What System Logger is, the idea of log streams and connectors, what structures, staging data sets, and offload data sets are used for, and how offload processing works. Explain the differences between DASDONLY and CF log streams. Stress that log stream data sets must *never* be manually deleted. Any deletions must be done using DELETE LOGSTREAM commands in an IXCMIAPU job. Explain that all access to data in log streams is via calls to the System Logger address space. Show the operators how to use the various DISPLAY LOGGER console commands to obtain information about defined log streams and how they are being used.

**Note:** It is also important to note that you must *never* delete log stream data sets using IDCAMS or IEHPRGM utilities, as this might delete valid data and cause problems for the users of that data when they attempt to reference it.

- ▶ The interaction between SMF and Logger. Explain dataset versus logstream recording modes, and how logstream mode means that they will no longer see messages about the SYS1.MAN data sets filling and switching. Show how different SMF record types are sent to different log streams, and that each log stream has its own set of buffers. Explain that log stream mode uses IFASMF DL instead of IFASMF DP. Show how the recording mode can be changed using the **SETSMF RECORDING** command or by swapping between SMFPRMxx members using the **SET SMF=xx** command. Show the various D SMF commands and how they are affected by the move to logstream mode.



## 4.2.8 Automation considerations

Because SMF logstream mode is a fundamental rewrite of the backend part of SMF, most of the messages related to logstream mode are new, which means that they probably are not defined to your automation product. However, because your SMF data is so important to your enterprise, it is vital that any messages about potential problems related to the SMF log streams are investigated and acted upon as quickly as possible. To achieve this, it is vital that these messages are defined to automation and that, at a minimum, automation will bring them to the attention of the appropriate staff. As you gain more experience with SMF logstream mode, you might enhance your automation by getting it to automatically respond to, and correct, some of the problems described by the messages. But for now, our primary concern is to ensure that important messages do not go unnoticed.

In this section we provide a set of messages that we encountered during our project and that we feel are important that someone sees. However, this is *not* a comprehensive set of all the messages that are important in an SMF logstream mode environment. We strongly suggest that you review all the messages starting with IFA7\* and IFA8\* in *z/OS MVS System Messages Vol8*, SA22-7638, and determine which must be defined to your automation product. You will also see that we have listed messages other than the ones that are directly issued by SMF. These other messages might relate to different aspects of system activity or system problems that can impact SMF, so we believe that these must also be monitored.

### SMF messages

The messages that we encountered are as follows. Add these messages to your automation product:

- ▶ IFA702I NO LOGSTREAMS WERE SPECIFIED FOR THE FOLLOWING RECORD TYPES n1-nx  
A SET SMF command was issued, but the system found that some SMF records have no assigned log stream in SMFPRMxx.
- ▶ IFA707I  
SMF cannot connect to the logstream.
- ▶ IFA710I LOGSTREAM PARAMETERS WILL NOT BE USED DUE TO ERROR  
This is issued when a SET SMF command did not complete.
- ▶ IFA713I SMF DATA LOST text  
This is issued when there is no log stream available at IPL, or when SMF has filled up all the buffers in the SMF address space.
- ▶ IFA717I LOGSTREAMS ARE NOT USABLE BY SMF. DATA BEING BUFFERED TIME=hh.mm.ss  
This is issued together with IFA710I when a SET SMF command did not complete.
- ▶ IFA718E  
Resource unavailable, data being buffered.
- ▶ IFA719I  
Log streams were disconnected and buffered data was lost.
- ▶ IFA721E  
SMF stopped due to ABEND. Issue **SET SMF=xx** to restart.
- ▶ IFA724E  
Unable to write to log stream. Data is being buffered.

- ▶ IFA785E SMF HAS USED nn OF AVAILABLE BUFFER SPACE FOR LOGSTREAM  
lsname

This is issued for BUFUSEWARN, as buffer usage changes once the specified threshold is reached. This is equivalent to IEE986E when in dataset mode. This message is only presented for z/OS 1.12 and later.

- ▶ IFA786W SMF DATA LOST - NO BUFFER SPACE AVAILABLE TIME=hh.mm.ss FOR  
LOGSTREAM lsname

This is issued when NOBUFFS(MSG) is specified for a log stream. This message takes the place of existing message IFA713I and is equivalent to IEE979W when in dataset mode.

- ▶ IFA787E SYSTEM WAIT STATE 'D0D-02'X - NO SMF BUFFERS FOR LOGSTREAM  
lsname

This is issued with a restartable wait state x'D0D' when NOBUFFS(HALT) is specified for a log stream. This message is equivalent to IEE987E when in dataset mode.

## System Logger messages

There are also messages issued by System Logger that indicate problems or potential problems with the SMF log streams. These should also be added to your automation product (but keep in mind that these messages can be issued for any log stream, not only the SMF ones):

- ▶ IXG114A OFFLOAD IS NOT PROGRESSING ON sysname LOGSTREAM: logstream,  
STRUCTURE: strname CHECK FOR OFFLOAD INHIBITORS WITHIN checksys
- ▶ IXG115A CORRECT THE OFFLOAD CONDITION ON sysname FOR strname OR  
REPLY TASK=END TO END THE STRUCTURE TASK
- ▶ IXG257I - DATA SET DIRECTORY FOR LOGSTREAM logstream IN STRUCTURE  
strname IN GROUP group IS OVER 90% FULL
- ▶ IXG261E - SHORTAGE OF DIRECTORY EXTENT RECORDS TOTAL numTotal IN USE:  
numInuse AVAILABLE: numAvail
- ▶ IXG262A - CRITICAL SHORTAGE OF DIRECTORY EXTENT RECORDS TOTAL  
numTotal IN USE: numInuse AVAILABLE: numAvail
- ▶ IXG272E LOGGER group taskname TASK DELAYED, REPLY "MONITOR", "IGNORE",  
"FAIL", "EXIT"
- ▶ IXG271I LOGGER DATA SET REQUEST IN group taskname SERVICE TASK DELAYED  
DURING THE PAST seconds SECONDS FOR LOGSTREAM logstream staging  
DSN=dsname, DIAG=diag
- ▶ IXG281I IXGLOGR DATA SET RECALL REQUESTS PENDING FOR GROUP:group
- ▶ IXG301I SYSTEM LOGGER FAILED TO OFFLOAD DATA FOR LOG STREAM logstream  
IN STRUCTURE strname  
  
If the return code is X'08' and the reason code is X'085', an offload had failed because of a directory full condition.
- ▶ IXG310I SYSTEM LOGGER CURRENT OFFLOAD IS NOT PROGRESSING FOR  
LOGSTREAM logstream STRUCTURE: strname request DSN=dsnhlq.dsnlsn.dsnllq
- ▶ IXG311I SYSTEM LOGGER CURRENT OFFLOAD HAS NOT PROGRESSED DURING  
THE PAST seconds SECONDS FOR LOGSTREAM logstream STRUCTURE: strname  
request DSN=dsnhlq.dsnlsn.dsnllq
- ▶ IXG312E OFFLOAD DELAYED FOR logstream, REPLY "MONITOR", "IGNORE", "FAIL",  
"AUTOFAIL", OR "EXIT"

This is not a comprehensive list of Logger messages of which to be aware. For more information about important Logger messages, refer to Appendix A, “Important Sysplex Messages,” in the IBM Redbooks publication *System z Parallel Sysplex Best Practices*, SG24-7817.

## Messages when switching between logstream and dataset mode

When you switch from SMF dataset to logstream mode (using either the **SETSMF** or **SET SMF** command), the IEFU29 exit is invoked. This ensures that any SMF data that was in the SYS1.MAN data set that was in use at the time of the switch is archived to wherever the SMF records are moved when the SYS1.MAN data sets fill. So, when you move from dataset to logstream mode, you do not need to take any manual action to save the SMF data.

However, if you switch from logstream mode back to dataset mode, the corresponding IEFU29L exit is *not* invoked. This means that you now have SMF data sitting in the log streams that were in use just prior to the switch. You can, of course, submit a job manually to extract and save all that data. However, to reduce the possibility of human error, it might be prudent to add automation that will trap the messages indicating the move back to dataset mode, and have that automation initiate whatever processes are required to save that data.

**Note:** If your system does *not* have the PTF for APAR OA34589 installed, you will not be able to use IFASMF DL ARCHIVE to archive all the log blocks from the log streams. In that case, use IFASMF DL DUMP to retrieve all the SMF records from the log streams, followed by deleting and redefining the log streams.

## System shutdown

Another consideration is related to the **Z EOD** command, typically issued during shutdown processing.

**Note:** Always issue the **Z EOD** command as part of the process of shutting down a system to avoid any data loss of SMF data, regardless of the SMF recording mode.

In dataset mode, the following actions take place:

- ▶ A type 19 record is created for each online direct access device (if the type 19 records are enabled in the SMFPRMxx Parmlib member).
- ▶ Empty the SMF buffers to the current active SMF data set.
- ▶ Switch to the next available SMF data set to allow the previous active data sets to be dumped.
- ▶ Issue the IEE334I HALT EOD SUCCESSFUL message.

When in logstream mode, the following actions take place:

- ▶ SMF empties all its buffers to the corresponding log streams.
- ▶ Up to z/OS V1R11, SMF disconnects from the log stream. Starting with z/OS 1.12, SMF no longer disconnects from the log stream as part of Z EOD processing.
- ▶ If the SYS1.MAN data sets are still defined in the active SMFPRMxx member, the IEFU29 exit is invoked to ensure that those data sets are emptied and the data captured.  
Note that the IEFU29L exit is *not* invoked.
- ▶ Issue message IEE334I HALT EOD SUCCESSFUL.

However, in logstream mode, the corresponding messages (IEE334I and IFA705I) only indicate that the SMF buffers have all been written to the log stream. It is possible, even likely,

that when that message is issued, the records are still in interim storage and have not been moved to the offload data sets yet. Furthermore, starting with z/OS 1.12, SMF no longer disconnects from its log streams as a result of the Z EOD command, and without that disconnect, there is no trigger to System Logger during system shutdown that it should move everything from interim storage to the offload data sets.

There are two situations to consider for each log stream:

- The log stream, or the structure the log stream resides in, is being used by more than one system.

Even though SMF does not disconnect from the log stream when the Z EOD command is issued, when the system that is being shut down is partitioned from the sysplex, one of the other systems that is connected to the same log stream or the same structure will do an offload and clean up the connection from the system that was shutdown.

This is one of the reasons why we suggest that each log stream be used by multiple systems.

- The log stream, or the structure the log stream resides in, is only being used by one system.

In this case, the data in the log stream remains in the log stream until that system connects to the log stream again. The only real exposure is for a CF-only log stream, and even in that case, the only exposure is if the CF goes down before the system reconnects to the log stream.

If this is a concern to you, you can manually initiate an offload by running the IBM-provided IXGOFLDS process on another member of the sysplex. If the availability of all your SMF data is critically important to you, you might integrate the start of this process (for each SMF log stream) on another member of the sysplex as part of your shutdown automation process. Example 4-6 shows an example of the use of IXGOFLDS.

*Example 4-6 Use of IXGOFLDS process*

---

```

S IXGOFLDS,LOGSTRM=IFASMF.#@$#PLEX.TYPALL
$HASP100 IXGOFLDS ON STCINRDR
IEF695I START IXGOFLDS WITH JOBNAME IXGOFLDS IS ASSIGNED TO USER STC
      , GROUP SYS1
$HASP373 IXGOFLDS STARTED
IXC582I STRUCTURE IFASMF_TYPALL ALLOCATED BY SIZE/RATIOS. 929
  PHYSICAL STRUCTURE VERSION: C705D47F 7D361B32
  STRUCTURE TYPE:                LIST
  CFNAME:                        FACIL03
...
IXL014I IXLCONN REQUEST FOR STRUCTURE IFASMF_TYPALL 930
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 0017
CONNECTOR NAME: IXGLOGR_#@$3 CFNAME: FACIL03
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 931
STRUCTURE IFASMF_TYPALL, CONNECTOR NAME IXGLOGR_#@$3
  CFNAME      ALLOCATION STATUS/FAILURE REASON
  -----
  FACIL03     STRUCTURE ALLOCATED AC001800
  FACIL04     PREFERRED CF ALREADY SELECTED A8001800
IXG283I STAGING DATASET IXGLOGR.IFASMF.#@$#PLEX.TYPALL.#@$3 937
ALLOCATED NEW FOR LOGSTREAM IFASMF.#@$#PLEX.TYPALL
CISIZE=4K, SIZE=41287680
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS01456
+IXG273I OFFLOAD COMPLETED SUCCESSFULLY

```

---

## SMS messages

If you SMS-manage your staging and offload data sets, SMS issues the following message to warn you if the storage group utilization has exceeded the specified threshold:

```
IGD17380I STORAGE GROUP (sgname) IS ESTIMATED AT xx% OF CAPACITY, WHICH EXCEEDS  
ITS HIGH ALLOCATION THRESHOLD OF zz%
```

Remember that these messages might be issued for any storage group, not only the ones used by SMF log streams. It is prudent to check whether this message is defined to your automation product, and if so, what actions are taken. Regardless, ensure that automation will make sure that someone who understands the urgency of the situation is made aware if this message is issued for the storage group that contains the SMF log stream data sets.

## Catalog message

The catalog address space monitors the percent of the maximum number of extents being used by each catalog against a defined threshold. The default threshold is 80%, but this value can be overridden by issuing a **F CATALOG,NOTIFYEXTENT(xx)** command.

If the space usage within a catalog exceeds the threshold, an IEC361I message will be issued, providing the name of the catalog and the percent of maximum extents that are currently in use. At a minimum, automation should be set up to monitor for this message and checked to see whether the named catalog is the one that contains the SMF log stream offload data sets. If it is, the appropriate personnel should be notified.

## Monitoring SMF activity

When running in log stream mode, the frequency at which the SMF data set full message is issued can be used to gauge the level of SMF activity. If these messages are normally issued every 30 minutes in peak times, and suddenly they are being issued every 5 minutes, that is an indicator of a potential problem.

When you are running in logstream mode, there are no SYS1.MAN data sets to fill, and therefore no messages indicating that the data sets have switched. However, if you still want to have a mechanism to monitor for unusually high levels of SMF activity, you can write automation to monitor for IXG283I messages that are issued every time that Logger allocates a new offload or staging data set. The message contains the log stream name. If you want to, you can write automation that will trap these messages, check the log stream name and the time that the last IXG283I message was issued for that log stream, and issue an alert if the frequency is above a level that you deem to be acceptable.

If you do write automation to process the IXG283I messages, it is worthwhile to check the CISIZE (which is also provided on that message) and inform the system programmer responsible for SMF if any SMF log stream offload data sets have a CISIZE other than 24 K.

## Other sysplex-related messages

In addition to the messages listed here, there are other messages that relate to the general health of your sysplex. Some of these are listed in an appendix in the IBM Redbooks publication *System z Parallel Sysplex Best Practices*, SG24-7817. Review the messages listed in that document and ensure that they are all defined to your automation product.

### 4.2.9 Install all SMF logstream mode-related service

At the time of writing, SMF logstream mode is still relatively new, and many enhancements are still being introduced, sometimes by PTF, and sometimes they are rolled back to previous releases of z/OS.

Before you start implementing logstream mode, it is prudent to retrieve and apply the latest logstream mode-related service. The easiest way to do this is to retrieve and install the latest FULL Enhanced HOLDDATA. This can be obtained from the web at:

<http://service.software.ibm.com/holdata/390holddata.html>

Ensure that you retrieve the package called FULL. That is the only one that contains the FIXCAT information that you will need to identify all logstream-related service.

After you have received that HOLDDATA into SMP/E, run an SMP/E job using the control statements shown in Example 4-7. Note that there is a dedicated fix category for SMF called IBM.FUNCTION.SMFlogstream. This fix category name must be specified when you run the REPORT MISSINGFIX job.

*Example 4-7 SMP control statements to identify missing PTFs*

---

```
//SMPCTL DD *
SET    BOUNDARY (GLOBAL)
      .
REPORT
      MISSINGFIX
      ZONES (
            MVSTD00
            MVSTD01
          )
      FIXCAT(
            IBM.FUNCTION.SMFlogstream
          )
      .
```

---

This job produces a report similar to that shown in Figure 4-8.

MISSING FIXCAT SYSMOD REPORT FOR ZONE MVSTA00							
FIX CATEGORY	FMID	HOLD CLASS	MISSING APAR	HELD SYSMOD	RESOLVING SYSMOD		
					NAME	STATUS	RECEIVED
IBM.Function.SMFLogstream							
	HBB7750		AA29743	HBB7750	UA50700	GOOD	NO
			AA29903	HBB7750	UA49815	GOOD	YES
					UA50172	GOOD	YES
					UA50716	HELD	YES
			AA30297	HBB7750	UA50172	GOOD	YES
					UA50716	HELD	YES
			AA30318	HBB7750	UA50716	HELD	YES
			AA30548	HBB7750	UA52444	GOOD	NO
			AA30848	HBB7750	UA54052	GOOD	YES
			AA30950	HBB7750	UA52040	GOOD	YES
			AA30974	HBB7750	UA52144	GOOD	YES
					UA55696	GOOD	YES
			AA31037	HBB7750	UA51935	GOOD	YES
					UA51981	HELD	YES
			AA31461	HBB7750	UA52328	GOOD	NO
			AA31737	HBB7750	UA55964	GOOD	YES
			AA31825	HBB7750	UA52619	GOOD	YES
					UA55964	GOOD	YES
			AA32632	HBB7750	UA54546	GOOD	YES
			AA32921	HBB7750	UA56146	GOOD	YES
			AA33244	HBB7750	UA55696	GOOD	YES

Figure 4-8 Sample REPORT MISSINGFIX report

The report identifies SMF logstream mode-related PTFs that are available but not applied to the target zone that you specified in the REPORT MISSINGFIX job. The job also creates a set of SMP/E control statements that will retrieve the fixes that have not yet been RECEIVED into SMP/E, followed by an APPLY CHECK of all the PTFs that are not held. Example 4-8 shows a sample of these statements.

Example 4-8 SMP/E control statements generated by REPORT MISSINGFIX job

```

RECEIVE ORDER(
    CONTENT(ALL) /*

    SMP/E RECOMMENDS ORDERING AND RECEIVING ALL APPLICABLE
    PTFs. IF YOU CHOOSE NOT TO ORDER ALL, THEN ORDER ONLY
    THE RESOLVING PTFs:

    CONTENT(PTFS(
        UA50700 UA52328 UA52444
    ) )
    */
    ORDERSERVER(SMPOSRVR) /* SPECIFY THE ORDERSERVER DDNAME. */
    CLIENT (SMPCLNT ) /* SPECIFY THE CLIENT DDNAME. */
)
DELETEPKG.
SET BDY(MVSTA00 ).
APPLY CHECK
SELECT(

```

```

/* IBM.Function.SMFLogstream                                */
    UA49815
    UA50172
    UA50700
/* UA50716 */
    UA51935
/* UA51981 */
    UA52040
    UA52144
    UA52328
    UA52444
    UA52619
    UA54052
    UA54546
    UA55696
    UA55964
    UA56146
)
GROUPEXTEND.

```

---

Ensure that the systems that you will be migrating to SMF logstream mode are as up to date with SMF logstream-related service as possible.

#### 4.2.10 System programmer SMF toolkit

In preparation for your migration to logstream mode, we strongly urge you to do extensive testing with SMF log streams on a test system. Specifically, familiarize yourself with all the new options on the IFASMFDL program and with the use of the LOGR subsystem interface with the SMF-supplied IFASEXIT program.

As part of your testing, you need to understand exactly what SMF records are in the input file or log stream, and which SMF records are in the output files. Some of the tools that you might use in this testing might be new to you, so we document them here.



## Getting a list of defined SMF log streams

The most basic requirement is to get a list of SMF log streams that are currently defined to Logger. The fastest way to do this is to issue a `D LOGGER,L,LSN=IFASMF.*` command. This provides you with a list of all the SMF log streams, whether they are CF log streams or DASDONLY, and whether they are currently in use. Figure 4-9 shows a sample invocation.

```
D LOGGER,L,LSN=IFASMF.*
IXG601I  18.03.05  LOGGER DISPLAY 800
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM                STRUCTURE                #CONN  STATUS
-----
IFASMF.##$#PLEX.TYPALL    IFASMF_TYPALL    000000  AVAILABLE
IFASMF.##$#PLEX.TYPRMF    IFASMF_TYPRMF    000001  IN USE
  SYSNAME: ##$2
    DUPLEXING: LOCAL BUFFERS
    GROUP: PRODUCTION
IFASMF.STRIPE.TYPDFLT      *DASDONLY*              000001  IN USE
  SYSNAME: ##$2
    DUPLEXING: STAGING DATA SET
    GROUP: PRODUCTION

NUMBER OF LOGSTREAMS:  000003
```

Figure 4-9 Using `D LOGGER` command to obtain list of SMF log streams

In this example, you will see that three SMF log streams are defined. Currently, only two of them are in use, and one of those is a DASDONLY log stream. The output also identifies which systems are currently connected to each log stream.

## Displaying range of dates in each log stream

Having obtained a list of log streams, you are likely to want to know what range of dates is contained in the log stream. You will especially need this information to carry out testing of the `DUMP`, `DELETE`, and `ARCHIVE` options of `IFASMF DL`, as well as the use of `RELATIVEDATE`, `SMARTENDPOINT`, and the `IFASEXIT` program.

Example 4-9 shows a sample JCL. A sample job is also provided in member `LISTLS` in the `RBITSO.CNTL` data set that is provided in Appendix E, “Additional material” on page 135.

Example 4-9 Sample job to list log stream date ranges

```
//KYNEFI JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(NO)
  LIST LOGSTREAM NAME(IFASMF.##$#PLEX.TYPRMF) DETAIL(YES)
```

This JCL produces a report like that shown in Figure 4-10.

DATA SET NAMES IN USE: IXGLOGR.IFASMF.##\$#PLEX.TYPRMF.<SEQ#>					
Ext.	<SEQ#>	Lowest Blockid / Highest Blockid	Highest GMT / Highest RBA	Highest Local / System Name	Status
-----					
*00001	A0000000	0000000000000001 00000000186DF8ED	12/05/10 03:30:00 186EDFA0	12/04/10 22:30:00 ##\$2	
	A0000001	00000000186EDFA1 0000000030DC31A1	12/05/10 08:28:00 186E51AC	12/05/10 03:28:00 ##\$2	
	A0000002	0000000030DD314D 00000000494B5D61	12/05/10 13:25:00 186EF418	12/05/10 08:25:00 ##\$2	

Figure 4-10 IXCMIAPU report showing range of dates in a log stream

In the report, you can see the range of dates in each offload data set. To understand the difference between DUMP and ARCHIVE, you might run a DUMP job with a start time that is later than the oldest data in the log stream. Then run the same JCL again, but using ARCHIVE instead of DUMP. You will see that a different number of SMF records is written to the output data set. After the ARCHIVE, run the IXCMIAPU job again to see which range of dates is still in the log stream (but note that no-longer-needed offload data sets are only deleted when Logger allocates a new offload data set for that log stream).

### Range of data in IFASMFDL and IFASMFDL files

While the IXCMIAPU job tells you which dates are in each log stream, it does not provide any information about which systems those records came from, or which specific SMF record types are in the log stream.

There are a number of options for obtaining this information, in various levels of detail, but perhaps the easiest to use is a set of IFASMFDL/IFASMFDL exits that are provided in Appendix E, “Additional material” on page 135. Information about the use of the exits, and samples of the reports that they provide, are contained in “Sample IFASMFDL exits” on page 147. But, briefly, the exits provide a report showing information similar to the IFASMFDL report, but expanded to include information at the system level and the SMF record subtype level. The reports also show the range of dates for each subtype that was found in the input file or written to the output file. The exits also have the ability to create a user SMF record containing the same information. These exits are useful to provide a quick check of whether an IFASMFDL or IFASMFDL job did what you expected it to do.

Before you can use these exits, you must define them in your SMFPRMxx member, as discussed in “Define IFASMFDL/IFASMFDL dump exits” on page 66.

### Printing contents of a data set containing SMF records

If you need more detail about the exact contents of a log stream or a sequential data set containing SMF records, there are a number of options, which we discuss in this section.

## IEBGENER

One simple option is to use IEBGENER to copy the contents to SYSOUT. Example 4-10 shows a sample job to do this for a sequential data set. The JCL for this sample is contained in member GENERSEQ in the RBITSO.CNTL data set in Appendix E, “Additional material” on page 135.

### *Example 4-10 Using IEBGENER to print SMF sequential data set*

---

```
//GENERSEQ JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=#0$2
//*****
//*
/* PRINT SEQUENTIAL DATA SET CONTAINING SMF RECORDS
//*
//*****
//STEP1 EXEC PGM=IEBGENER
//SYSUT1 DD DSN=ZS9037.Z#0$2.SMF DUMP(-1),DISP=SHR
//SYSUT2 DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
```

---

Example 4-11 shows an equivalent job to print the contents of an SMF log stream. The JCL for this sample is contained in member GENERLS in the RBITSO.CNTL data set in Appendix E, “Additional material” on page 135.

### *Example 4-11 Using IEBGENER to print contents of an SMF log stream*

---

```
//GENERLS JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM SYSAFF=#0$2
//*****
//*
/* PRINT LOG STREAM CONTENTS
//*
//*****
//STEP1 EXEC PGM=IEBGENER
/* CHANGE LOGSTREAMNAME TO REQUESTED LOG STREAM NAME
//SYSUT1 DD DSN=IFASMF.STRIPE.TYPDFLT,DISP=SHR,
// DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756),
// SUBSYS=(LOGR,IFASEXIT,'FROM=(2010/338,10:10),TO=(2010/338,19:40),X
// LOCAL')
//SYSUT2 DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
```

---

You will notice that this job uses the IFASEXIT program to extract the SMF records from the log stream. If you look at the resulting SYSOUT, you will see that each SMF record is presented as a separate line. This is because the IFASEXIT program understands the layout of the SMF records in the log stream and presents them to IEBGENER as separate records.

You will also notice that the GENERLS job uses options on the LOGR subsystem interface to specify a range of dates that should be extracted from the log stream. Be aware that the default is that any times that you specify are interpreted as GMT. If you want to use local times, you must specify “LOCAL” in the parameters, as shown in the example.

One other thing that you must be aware of is the importance of specifying the correct blocksize on the DD statement that points at the log stream. The blocksize determines the

length of the records that will be passed back to the calling program (IEBGENER in this case). If you specify a blocksize of 4096, all SMF records passed back to IEBGENER are truncated. Therefore, to ensure that you get the full SMF records, specify a blocksize of 32760.

**Note:** You can also use IEBGENER together with IFASEXIT to extract SMF records from a log stream into a sequential data set that can then be processed by IFASMFDP or another program. Note, however, that this is probably a less efficient mechanism than using IFASMDFL to achieve the same net result.

### IDCAMS PRINT

You can also use the IDCAMS PRINT facility to print the contents of a data set or a log stream (Example 4-12). You can see that this is basically the same as the IEBGENER example. If you want to print the contents of an SMF log stream, you simply point the input DD statement at the log stream and use the LOGR subsystem and IFASEXIT to format and return the records to IDCAMS.

Example 4-12 Using IDCAMS PRINT to print SMF records

```
//IDPRINTL JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*****
//*
//* PRINT LOGREC LOG STREAM CONTENTS
//*
//*****
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INFILE DD DSN=IFASMF.##$#PLEX.TYPRMF,
// DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756),
// SUBSYS=(LOGR,IFASEXIT,'FROM=OLDEST,TO=YOUNGEST')
//SYSIN DD *
PRINT INFILE(INFILE) COUNT(9)
/*
```

One advantage of IDCAMS PRINT compared to IEBGENER is that the output shows the content of the SMF records in hexadecimal and character format (Figure 4-11). If you need to see the date and times of the SMF records, this is vital. IEBGENER only shows the character format of the SMF records, so any binary fields will be displayed as blanks.

IDCAMS	SYSTEM SERVICES	TIME: 18:22:17	12/09/10	PAGE	2
LISTING OF DATA SET - IFASMF.##\$#PLEX.TYPRMF					
RECORD	SEQUENCE	NUMBER	-	1	
000000	DF460060	67B10110 338F7B7C 5BF2D9D4	C6400001 00070000 00000054 00680001	*...-.....#@\$2RMF	.....*
000020	000000BC	00DC0001 00000198 00480006	00000348 02AC0001 000005F4 00480021	*.....4.....*	
000040	00000F3C	005000AC 000044FC 00180006	760FD9D4 C6404040 40400173 200F0110	*.....&.....RMF	.....*
000060	338F0100	000F0000 0000003C 00001000	40404040 0001000F E9E5F0F1 F1F1F0F0	*.....ZV011100*	
000080	037E1C5B	C6FAD857 B3F00000 FFFFBCF1	DCC00000 00000000 00000000 003C0000	*.=.\$F.Q..0....1.{.....*	
0000A0	C6FAD857	B3F00000 7B7C5B7B D7D3C5E7	7B7C5BF2 40404040 28170020 18990000	*F.Q..0...#@\$#PLEX#@\$2	.....*
0000C0	F7F1F640	40404040 40404040 40404040	000C0002 00000214 00000032 00000002	*716	.....*
0000E0	00000000	00000000 D4F3F240 40404040	40404040 40404040 0002C6DB 03E13B59	*.....M32	..F.....*

Figure 4-11 Sample output from IDCAMS PRINT command

### Unformatted print of an SMF log stream

There might be times, possibly for debugging purposes, when you need to see the raw contents of a log stream (that is, you do not want the contents formatted into SMF records). This can be achieved by using the IXCSEXIT instead of the IFASEXIT program. IFASEXIT

obtains the log blocks from Logger and presents them to the calling program as SMF records. IXGSEXIT presents the log blocks exactly as they were read from the log stream, including any control information that IFASEXIT might use to format the records.

However, even though we thought that we might need this capability, in reality we never had a need to use it during our project.

### ***Accessing logically deleted SMF records***

If you specify a non-zero RETPD for a log stream, log blocks will not be physically deleted from the log stream until that retention period expires. For *normal* accesses to the log stream, it will appear that the data has been deleted. However, the SMF Exit IFASEXIT provides the ability to access log blocks that have been logically deleted, but not yet physically deleted. By default, IFASEXIT will return just the non-deleted records (referred to as ACTIVE records). However, you can use the INACTIVE|ACTIVE|ALL keyword to control whether non-deleted, deleted, or all records are to be returned to the calling program. This is described in detail in *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

### **Other functions using SMF log streams**

In addition to viewing the contents of an SMF log stream, there are other operations that you might want to carry out against a log stream containing SMF data, and a number of ways to achieve what you are trying to do.

### ***RMF postprocessor reports***

The RMF ISPF front end has been enhanced to support the ability to produce RMF reports using SMF records from log streams. As shown in Figure 4-12, the RMF ISPF dialog now gives you the option to specify SMFLOG as a source for the input data.

RMF - Postprocessor Setup

Command ==>

Input Data

==> SMFLOG\_

DATASET, SDS (Sysplex Data Server Buffers)  
or SMFLOG (SMF Log Streams)

Output Data

==> NO\_

YES or NO (NO to route output to SYSOUT)

Report Profile

==> \_\_\_\_\_

Edit generated JCL

==> NO\_

YES or NO

Job Statement Information:

==> //KLAEYP JOB (ACCOUNT),'PGMRNAME',CLASS=A,REGION=32M

==> /\*\*

==> /\*\*

==> /\*\*

Complete this panel and press ENTER to continue, or END to exit.

To return to RMF Primary Menu without saving input, enter CANCEL.

Figure 4-12 RMF postprocessing panel

## Using IFASEXIT with utility programs

The following examples show how easy it is to read the data from the log stream using IFASEXIT.

If you want to access active and logically deleted data in the log stream, use the VIEW=ALL and FROM=OLDEST,TO=YOUNGEST options on the SUBSYS parameter. In this example, we decided to keep the data for additional post processing and copied it to a VBS data set, which can then be accessed by IFASMFDP or other programs enabled to read SMF dump format data.

We can access log streams with IEBGENER (Example 4-13).

*Example 4-13 Using IEBGENER and LOGR subsystem to access log streams*

---

```
//*****  
/* Function: Read SMF records with IEBGENER from a log stream  
//*****  
//GENER      EXEC      PGM=IEBGENER  
//SYSUT1 DD   DISP=SHR,DSN=IFASMF.SYS1.TYPRMF,  
//   SUBSYS=(LOGR,IFASEXIT,'FROM=OLDEST,TO=YOUNGEST,VIEW=ALL'),  
//   DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756)  
//SYSUT2 DD   DSN=SMFRECS.IEBGENR.SMFDATA,SPACE=(CYL,(500,500)),  
//   UNIT=SYSDA,DCB=(RECFM=VBS,LRECL=32760),  
//   DISP=(NEW,CATLG,DELETE)  
//SYSPRINT DD   SYSOUT=*  
//SYSIN DD   DUMMY  
/*
```

---

After the successful completion of the IEBGENER step, we wanted to look at which data was copied from our log stream. We ran IFASMFDP against the previously created output data set (Example 4-14) to get a report showing the SMF record types that were written to the sequential data set.

*Example 4-14 IFASMFDP reads IEBGENER output*

---

```
//*****  
/* Function: Dump SMF records from data set  
//*****  
//STEP1      EXEC      PGM=IFASMFDP  
//SYSPRINT   DD        SYSOUT=*  
//EIN        DD        DISP=SHR,DSN=SMFRECS.IEBGENR.SMFDATA  
//AUS        DD        DSN=SMFRECS.DUMP.SMFDATA,  
//           UNIT=SYSDA,  
//           SPACE=(CYL,(800,600)),  
//           DISP=(NEW,CATLG,DELETE)  
//SYSIN      DD        *  
              INDD(EIN,OPTIONS(DUMP))  
              OUTDD(AUS,TYPE(000:255))  
/*
```

---

Figure 4-13 shows the output from the IFASMFDP job. As you can see, the data set contains only the RMF record types 70 to 78, as you would expect. Note that the type 2 and type 3 records are always written by IFASMFDP and IFASMF DL to mark the start and end of a data set containing SMF data.

IFA010I SMF DUMP PARAMETERS IFA010I END(2400) -- DEFAULT IFA010I START(0000) -- DEFAULT IFA010I DATE(1900000,2099366) -- DEFAULT IFA010I OUTDD(AUS,TYPE(0:255)) -- SYSIN IFA010I INDD(EIN,OPTIONS(DUMP)) -- SYSIN IFA020I AUS -- SMFRECS.SMF DUMP.SMF DATA IFA020I EIN -- SMFRECS.IEBGENR.SMF DATA					
SUMMARY ACTIVITY REPORT					
START DATE-TIME	08/12/2010-09:30:00				
RECORD	RECORDS	PERCENT	AVG. RECORD	MIN. RECORD	END D
TYPE	READ	OF TOTAL	LENGTH	LENGTH	
2	0				
3	0				
70	54	1.36 %	11,472.00		852
71	27	.68 %	1,920.00		1,920
72	1,728	43.54 %	1,389.43		1,108
73	27	.68 %	21,728.00		21,728
74	1,971	49.66 %	19,756.84		364
75	81	2.04 %	264.00		264
77	27	.68 %	681.48		320
78	54	1.36 %	11,780.00		1,888
TOTAL	3,969	100 %	10,903.39		264
NUMBER OF RECORDS IN ERROR 0					

Figure 4-13 IFASMFDP report for IEBGENER-created data set

Our next example (Example 4-15) shows an IDCAMS step to copy RMF records from a log stream to a sequential data set.

Example 4-15 Use IDCAMS to access data in a log stream

```
//REPRO EXEC PGM=IDCAMS
//INDD DD DSN=IFASMF.#@$A.TYPRMF,
// DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756)
// SUBSYS=(LOGR,IFASEXIT,'FROM=OLDEST,TO=YOUNGEST,VIEW=INACTIVE')
//OUTDD DD DSN=SYS3.#@$A.TRMFVBS,DISP=(,CATLG,DELETE),
// SPACE=(CYL,(3500,3000)),UNIT=SYSDA,
// DCB=(RECFM=VBS,LRECL=32760)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
REPRO INFILE(INDD) OUTFILE(OUTDD)
/*
```

The last example (Example 4-16) shows how to use DFSORT to copy and sort the data from a log stream.

Example 4-16 DFSORT sample to copy and sort data from a log stream

```
//*****
//* RMF SORT PROCESSING
```

```

//*****
//RMFSORT2 EXEC PGM=SORT,REGION=0M
//SORTIN DD DSN=IFASMF.#@$#PLEX.ALL.CFONLY,
// DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756),
// SUBSYS=(LOGR,IFASEXIT,'FROM=(2010/224,12:00),TO=YOUNGEST')
//SORTOUT DD DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SORTWK01 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SORTWK02 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SORTWK03 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
MODS E15=(ERBPPE15,36000,,N),E35=(ERBPPE35,3000,,N)
/*

```

---

### 4.2.11 Post-implementation monitoring

After you move the first system to logstream mode, and as you move more and more systems over to that mode, verify that the performance of the log streams is in line with your expectations. More information about how this checking might be carried out is provided in 5.3.1, “Monitoring Logger activity” on page 107.

However, at this point we just want to ensure that you are collecting the information that will be required to enable your performance review. The SMF record types that you will be studying are:

- ▶ SMF Type 7 (SMF Activity - only supports logstream mode in z/OS 1.12 and later)
- ▶ SMF Type 70 to 79 (RMF)
- ▶ SMF 88 (System Logger)

It is a good idea to collect these records for a reasonable period of time before switching to logstream mode, so you have a sound basis for doing a comparison.

## 4.3 Planning for approach 2 implementation

Approach 2 is intended to help you exploit the throughput benefits of log stream mode, but with minimal changes to your SMF structure. Therefore, the amount of planning (above and beyond the topics that we have already covered) is minimal.

### 4.3.1 Document existing SMF job network

The fundamental change between dataset mode and approach 2 is that the SYS1.MAN data sets will be replaced with one or more log streams. Therefore, to ensure a smooth migration, it is vital to ensure that you are aware of every program and job that accesses the SYS1.MAN data sets. Most likely, the only program that uses those data sets will be IFASMFDP. However, use the output from the SMFDSACC REXX (documented in “SMFDSACC program” on page 140) to identify all the jobs or started tasks that access those data sets. After you have the complete list you will have to document how each one will be affected by the change to logstream mode.



### 4.3.2 Changes to SMF jobs

Remember that approach 2 does not affect any SMF data sets beyond the data set that you empty your SYS1.MAN data sets into. So any jobs that run against any of those data set will be unchanged, and the jobs that empty the SYS1.MAN data sets should not be changed. When you convert to logstream mode, the event that triggers those jobs (the SYS1.MAN data sets filling) will no longer occur, meaning that those jobs will not be started. And if they are not going to be run, there is no need to change them. In fact, you specifically do *not* want to change them, so that if you want to fall back to dataset mode for any reason, you can be sure that everything that works today will continue to work at that time.

That does not mean that *no* changes are required. Given that approach 2 involves archiving all the SMF data to the existing sequential data sets on a regular basis, you will need to:

- ▶ Create jobs to archive the log streams.
- ▶ Implement a mechanism for submitting those jobs on a regular basis.

#### Sample archive jobs

You will recall that an IFASMF DL ARCHIVE or DELETE must delete all log blocks within the time range that it is asked to process. This means that if you want to extract SMF records from a log stream to more than one destination, all those records must be processed in the same invocation of IFASMF DL. Because IFASMF DL does not support the SID parameter on the OUTDD statement, you must move the records from all systems to a single data set, and you can then split the records out by system from that data set if that is your intention.

Example 4-17 shows a sample job that moves all the records from the IFASMF.PRODPLEX.PERF log stream. The second step in the job moves records to separate data sets for each system.

*Example 4-17 ARCHIVE job to process data for multiple systems*

---

```
//STEP1 EXEC PGM=IFASMF DL
//OUTDD1 DD DSN=SMF.#@$#PLEX.PERFRECS,DISP=(,CATLG,DELETE),
//          SPACE=(CYL,(200,200),RLSE),
//          DCB=(DSORG=PS,BUFNO=18,RECFM=VBS,LRECL=32760,BLKSIZE=0)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
LSNAME(IFASMF.PRODPLEX.PERF,OPTIONS(ARCHIVE))
OUTDD(OUTDD1,TYPE(0:255))
RELATIVEDATE(BYDAY,0,1)
/*
//ICET1 EXEC PGM=ICETOOL
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//TOOLIN DD *
COPY FROM(CIN) TO(SYSA) USING(CPY1)
/*
//CPY1CNTL DD *
OUTFIL FNames=SYSA,INCLUDE=(15,4,CH,EQ,C'SYSA')
OUTFIL FNames=SYSB,INCLUDE=(15,4,CH,EQ,C'SYSB')
//SMFIN DD DISP=SHR,
//          DSN=SMF.#@$#PLEX.PERFRECS
//SYSA DD DSN=SMF.SYSA.PERF.DAILY,
//          DISP=(,KEEP)
```

```
//SYSB      DD DSN=SMF.SYSB.PERF.DAILY,  
//          DISP=(NEW,KEEP)
```

---

The sample job in Example 4-17 on page 91 is for a sysplex with just two systems:

- ▶ SYSA
- ▶ SYSB

If there were more systems in the sysplex, you would add more output data sets and corresponding DD statements and more OUTFIL control statements. The SMF.SYSx.PERF.DAILY data sets are the data sets that are used by the existing IFASMFDP jobs that empty the SYS1.MAN data sets when they fill. Because the SMFPRMxx member controls which SMF record types are placed in the log stream, the ARCHIVE job specifies TYPE(0:255). This means that if any new record types get added to this log stream in the future, you do not need to worry about changing the ARCHIVE job. Note also the use of RELATIVEDATE(BYDAY,0,1). The use of this parameter was discussed in 1.4, “SMF support for log streams” on page 9.

Assuming that you will have multiple log streams, it is better to have a separate job for each log stream. This gives you the ability to archive multiple log streams in parallel.

### Submitting archive jobs

You are most likely switching SYS1.MAN data sets many times a day at the moment, and each time you switch, you run an IFASMFDP job to empty the switched-from data set, to make it available for use again.

However, running the IFASMFDP job many times a day provides another benefit—the amount of SMF data processed by each run is limited, meaning that each job does not run for long. And this means that the data sets containing all of yesterday’s data are available for use soon after midnight.

When you switch to logstream mode, we suggest that you also run the archive jobs many times a day. The frequency depends on the volumes of SMF data that you have to process and how soon after midnight you need yesterday’s data. But also remember that System Logger has to process all the data that you are archiving. So, if you were to only run one archive a day, the elapsed time would likely be quite long, and the impact on System Logger would also go on for a long time. By running archive many times during the day, you reduce the elapsed time for each job and the impact on System Logger. Also attempt to schedule the archive of each log stream to run at a different time. Remember that, initially at least, you will probably be moving the records from all the log streams into your existing SMF offload sequential data set. By running the archives at different times, you will reduce the contention on that data set that you might otherwise experience.

You also need to consider how the jobs will be submitted. Generally, you have two options:

- ▶ The archive jobs can be kicked off directly or indirectly by automation.
- ▶ The archive jobs can be submitted by your batch workload scheduler.

We believe that the latter is the more desirable option. The reason for this is that your scheduler will track the execution of the job and raise an alert if the job does not complete successfully.

At this time, decide which mechanism you will use, and set up the JCL (either jobs or started tasks) that will do the archive for each log stream that you will be using.

### 4.3.3 Fallback planning

Should you need or want to fall back to dataset mode, the main consideration is to ensure that the data that was in the log streams prior to switching back to dataset mode is not lost. In approach 2, you will be moving the SMF records from the log streams to the SMF sequential data sets on a regular basis, and when you fall back to dataset mode, you will have a certain amount of time before the SYS1.MAN data set fills and you do the next offload. Ensure that you move all the SMF records from the log streams before the SYS1.MAN data set fills. The best way to do this is to put automation in place that will automatically submit a job to move all the SMF records. This automation was discussed in “Messages when switching between logstream and dataset mode” on page 77.

## 4.4 Planning for approach 3 implementation

At the start of approach 3, all your SMF data is being written to log streams and then shortly thereafter being moved to sequential data sets. Approach 3 involves changing your handling of selected log streams so that the data is kept in the log stream until it expires and is no longer moved to the sequential data sets.

Therefore, the activities that need to be planned for include:

- ▶ Changing the archive jobs so that the SMF records are no longer moved from the log stream.
- ▶ Update the log stream definition to indicate that Logger is now responsible for deleting the log blocks, and how long the data should be kept in the log stream before it is deleted.
- ▶ Providing users with a set of JCLs that will use the log stream as the source of SMF records, rather than the sequential data sets.
- ▶ Provide documentation so that users know which range of dates are in the sequential data sets and which range of dates are in the log stream.
- ▶ Provide education for the users of the SMF records that will now reside in the log stream.
- ▶ Devise a process to handle the expiration of old GDGs.
- ▶ Create a fallback plan.

**Note:** If you collect SMF Type 70 and 89 records for software charging purposes, those records must be gathered from every system in your installation into a single data set. Remember that a log stream can only contain data from one sysplex, so even if you decide to keep these record types in a log stream instead of archiving them to sequential data sets, you will still need to copy these records out of the log stream so that they can be merged with the corresponding records from all your other systems.

### 4.4.1 Archive job changes

When you are ready to move away from archiving the SMF records from a given log stream, leave the existing archive job exactly as it is (so that it will still be there should you need to back out the migration). The preferred option to stop the migration is simply to stop submitting the job or started task that does that archiving. This is another reason why we suggest using a separate archive job for each log stream: Not only does it provide more parallelism, it also makes it less disruptive when you want to stop archiving a given log stream.

## 4.4.2 Change log stream definitions

When you are archiving SMF records from a log stream to a data set, the IFASMFDDL program issues a delete command to System Logger to delete the log blocks that it has successfully archived from the log stream. So, in this case, you only want the log blocks to be deleted after they have successfully been archived.

However, when you move to keeping the data in the log stream, it makes sense to have Logger control the deletion of log blocks based on the retention period for that log stream. Therefore, when you are ready to stop archiving the log stream, you need to run a job to update the log stream definition to specify the actual number of days that you want the records to be kept in the log stream (on the RETPD keyword) and the fact that you now want Logger to do the deleting (on the AUTODELETE keyword).

## 4.4.3 Provide alternate JCL for processing log streams

If you have a job today that processes SMF data from a sequential data set and you decide to keep those SMF records in a log stream in the future, you will need to create a set of JCLs that will obtain the SMF records from the log stream instead of the sequential data set. But remember that there will be a period of time when records before the cutover date will continue to exist in the sequential data set, so do not discard that existing job. Rather, have two versions of that job's JCL—one that gets its data from the sequential data set, and the other that gets the data from the log stream.

For the version that gets the data from the log stream, you have two options:

- ▶ Insert an additional step in the job that uses IFASMFDDL DUMP to extract the required SMF records.
- ▶ Change the DD statement that currently points at the sequential data set to use the SUBSYS=LOGR interface with the IFASEXIT program.

At the time of writing, the first option is the usually most attractive, for the following reasons:

- ▶ In the IFASMFDDL control statements, you can use the RELATIVEDATE keyword. So if the job always creates a report for "yesterday," you can specify RELATIVEDATE(BYDAY,1,1) and run the job unchanged every day. If you use IFASEXIT, you need to change the JCL every day to specify a new date range.
- ▶ IFASMFDDL DUMP supports the SMARTENDPOINT keyword. The use of this keyword stops IFASMFDDL from reading all the way to the end of the log stream. This can have a dramatic (positive) impact on the elapsed time and resource usage for a job, especially if it is processing data from a long time ago.
- ▶ From the perspective of the user program, nothing has changed. Today, it reads a sequential data set, and in the future it would read a sequential data set (the one created by the new IFASMFDDL step). So no testing is required to ensure that the user program still works as before.

Example 4-18 shows a sample RMF job using IFASMFDDL DUMP. This job extracts all the records for yesterday from the sysplex-wide RMF log stream, sorts them into chronological order, and then passes them to the RMF postprocessor.

*Example 4-18 RMF job using IFASMFDDL to extract required SMF records*

---

```
//KYNEFDL JOB (0,0),'IFASMFDDL',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//STEP EXEC PGM=IFASMFDDL
//DUMPOUT DD DSN=KYNEF.RMF.DAILY,DISP=OLD
//SYSPRINT DD SYSOUT=*
```

```

//SYSIN      DD *
LSNAME(IFASMF.#@$#PLEX.TYPRMF,OPTIONS(DUMP))
OUTDD(DUMPOUT,TYPE(0:255))
RELATIVEDATE(BYDAY,1,1)
START(0900)
END(1200)
SMARTENDPOINT
USER3(SMFDPUX3)
USER2(SMFDPUX2)
USER1(SMFDPUX1)
//*****
//*          SORT RMF RECORDS
//*****
//RMFSORT2 EXEC PGM=SORT,REGION=OM
//SORTIN  DD  DISP=SHR,DSN=KYNEF.RMF.DAILY
//SORTOUT DD  DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,(5000,500))
//SORTWK01 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SORTWK02 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SORTWK03 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(2000,200))
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSIN    DD  *
        SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
        MODS E15=(ERBPPE15,36000,,N),E35=(ERBPPE35,3000,,N)
/*
/*
//*****
//*          RMF POSTPROCESSING
//*****
//RMFPP2 EXEC PGM=ERBRMFPP,REGION=OM
//MFPINPUT DD DISP=(OLD,DELETE),DSN=*.RMFSORT2.SORTOUT
//MFPMMSGDS DD  SYSOUT=*
//SYSIN    DD  *
        SUMMARY(INT,TOT)
        REPORTS(CPU,XCF)
        MAXPLEN(999)
        SYSOUT(X)
        DINTV(0001)
/*
//

```

---

### ***RELATIVEDATE option***

The RELATIVEDATE option allows you to easily select a range of records based on the current day. This parameter gives you the power to specify the unit of time (BYDAY, BYWEEK, BYMONTH), the number of units to move backwards, and the number of units to advance. So, for example, RELATIVEDATE(BYWEEK,2,1) will move back two weeks from the current week and advance one week.

An associated parameter is the WEEKSTART parameter. Since some locales start the week on a Sunday and some on a Monday, this keyword allows you to tailor RELATIVEDATE processing to match your understanding of the first day of the week.

## IFASEXIT

The attraction of using the IFASEXIT program is that it avoids the need to add an additional step to the job. Also, if the user program is going to extract a large amount of data from the log stream, it might be more efficient to use IFASEXIT and only process the data one time, rather than using IFASMFDL and processing the data twice (once to extract it to the temporary sequential data set, and a second time for the user program to read that data set). Another situation in which the use of IFASEXIT might be attractive is if the log stream only contains a small amount of data (for example, a log stream that is archived on an hourly basis). In that case, IFASEXIT and IFASMFDL would probably read the same amount of data from the log stream, so there is little or no savings to be had from the use of IFASMFDL.

If there are cases in which you determine that the preferred option is to use IFASEXIT, additional testing is required. There are restrictions that apply when using the SUBSYS=LOGR interface that do not apply when processing a sequential DASD data set:

- ▶ There is no way for a program reading from a log stream to obtain the DCB characteristics of the log stream (as is possible when reading a DASD data set), so the DCB (both the blocksize *and* the LRECL) must be explicitly specified for the log stream. If this information is not provided, various different types of abends might be encountered.
- ▶ There are actions that a program can carry out against a sequential data set that are not supported by the SUBSYS interface.

The full set of restrictions is documented in *z/OS MVS Assembler Services Guide*, SA22-7605, in the sections titled “Reading Data From Log Streams in Data Set Format” and “Is My Application Eligible for the LOGR Subsystem?”.

Remember that each generation of a GDG typically contains records from a finite time period, whereas a log stream contains log blocks from the oldest log block though to the most recently created ones. Therefore, JCL changes are required each time that the IFASEXIT program is run against the log stream to limit the time period for which data will be returned to the program.

### 4.4.4 Provide documentation about the location of SMF records by date

As you transition through a period in which there are still valid SMF records residing in the existing sequential data sets, and the more recent SMF records are in the log stream, it is helpful to your users if you can provide easily accessed information about the location of the data for a given range of dates. Also, because there will not be new generations of the GDG, the generation number of the GDG (if you are using GDGs) will no longer be relative to today's date. So, you might want to provide a table similar to Table 4-2.

Table 4-2 Sample table to locate SMF records

Relative generation	Date range (MM/DD/YY)	Date range (yy.ddd)
IFASMF.PRODPLEX.RMF	After 11/13/10	After 10.317
SMF.RMF.WKLY(-1)	11/07/10 - 11/13/10	10.311 - 10.317
SMF.RMF.WKLY(-2)	10/31/10 - 11/06/10	10.304 - 10.310
SMF.RMF.WKLY(-3)	10/24/10 - 10/30/10	10.297 - 10.303
SMF.RMF.WKLY(-4)	10/17/10 - 10/23/10	10.290 - 10.296

#### 4.4.5 SMF user education

Together with the sample JCL for extracting SMF records from a log stream, and the reference guide to show users the location of SMF records based on their creation date, it is valuable (and a good investment) to provide education for the users of the SMF records that you will now be keeping in a log stream. The Manage\_SMF spreadsheet should contain a list of the user IDs that run jobs against the SMF record types that are being affected by your change, allowing you to target the education at those users (and possibly even to tailor the education, depending on the users' level of experience with JCL and the sophistication of their use of SMF).

#### 4.4.6 Handling expired GDGs

If you are using a GDG today to contain the SMF records that will be affected by your change, the oldest generations are automatically deleted as you create new generations. For example, if you keep SMF data for two years in a monthly GDG, the GDG is probably defined to have 24 generations. So, when you create a new generation to hold last month's data, the GDG containing SMF records from 25 months ago will be deleted.

However, when you change so that the SMF records are no longer moved to the GDG, the oldest generation will no longer automatically be deleted. If you want, you can just leave it like that until you have two years' worth of data in the log stream, and then delete the GDG at that point. Or, you could create a process that will run every month and delete the oldest generation of the GDG. So, 23 months from now, there will only be one generation left, and then a month later, the last remaining generation would automatically be deleted at that point.

There is no correct approach to this. Which approach you take will reflect the requirements of your enterprise.

#### 4.4.7 Fallback plan

Given that the starting point for your approach 3 implementation is approach 2, the fallback in case of any problems with approach 3 is *not* to dataset mode. The fallback would be to how things were prior to the start of the move to approach 3 (that is, all SMF records are being archived from the log streams to sequential data sets).

While it is hoped that your migration to logstream mode will go smoothly, you need to be prepared in case you need to fall back for any reason. Therefore, you need to have a carefully thought-out, and well-tested fallback plan.

Apart from re-enabling the archive job and changing the log stream definitions so that the archive process will once again be responsible for deleting the records, and informing the users that they should once again retrieve the SMF data from the sequential data sets, the main consideration is what will happen with the data that now resides in the log stream.

If the decision to fall back is the result of encountering a problem, you presumably want all the data in that log stream to once again reside in a sequential data set. In that case, the next time an archive for that log stream is submitted, *all* the data in the log stream will be moved to the same sequential data set to which all the other log streams are being archived. This happens automatically (remember that an archive always starts at the beginning of the log stream). Depending on how long you have been running without doing an archive for that log stream, you need to be careful that the archive does not exhaust all the space available in the sequential data set. But other than that, everything should revert to the way that things were before.







# Implementation

At this stage, you have completed all the work that is required to prepare for a successful migration. This chapter discusses the steps to take when you are ready to move from System Management Facilities (SMF) dataset to logstream mode.

**Note:** The suggestions and descriptions in this chapter assume that the PTF for APAR OA34589 is applied.

If this PTF is not applied, refer to Appendix C, “Considerations for systems without APAR OA34589” on page 131, for information about how this might affect your implementation plans.

## 5.1 Preparation

The cutover to logstream mode consists of three phases:

- ▶ Getting all the definitions in place
- ▶ Making the actual switch to logstream mode
- ▶ Verifying that everything is working as expected

In this section we briefly cover the actions that are required to prepare for the switch and discuss the actions needed to complete the switch and subsequently verify that everything is working as you expect.

### 5.1.1 Adjusting LPAR storage amounts

In “Planning for processor storage requirements” on page 67, we discussed the potential storage requirements for running in logstream mode. If you determined that any of the LPARs that will be running SMF in logstream mode will require additional storage, that storage must be added at this time. As long as there is unassigned storage in your CPC, it is possible to add storage to a z/OS LPAR non-disruptively.

### 5.1.2 Preparing the SMS environment

In “Review SMS setup” on page 72, we discussed the considerations for the SMS constructs (data class, management class, and storage class) that will be used by the SMF log stream data sets. We also discussed the process of determining how much space will be required for the offload data sets, and which storage group should be used for those data sets.

If you identified a need to change existing class definitions or to set up new classes, that activity should be carried out at this point.

It is especially important to ensure that there is sufficient available storage in the storage group that will be used for the SMF log stream data sets. Remember to allow space for offload data sets that were migrated and subsequently need to be recalled, and ensure that the IGD17380I message is defined to your automation package.

### 5.1.3 Prepare the user catalogs

In “Ensure user catalog is large enough” on page 74, we discussed how the large number of offload data sets that might exist in SMF logstream mode can impact the user catalog that will contain those data sets.

If you determined that the catalog that you planned to use does not have sufficient free space, either select a new high-level qualifier for the SMF log stream data sets (and set up a new user catalog specifically for those data sets) or carry out whatever process you use to enlarge your existing catalogs. On balance, setting up a new HLQ and a new user catalog is likely to be easier. If you set up a new HLQ, remember to check that your SMS ACS routines will assign the intended SMS classes to the log stream data sets.

Also, this might be a good time to implement the suggestions about managing your catalog recovery SMF records as discussed in “Creating log stream definition statements” on page 61.

## 5.1.4 Implement automation changes

Section 4.2.8, “Automation considerations” on page 75, discussed the importance of defining critical messages to your automation product and making sure that, at a minimum, the appropriate staff (and their backup) is notified if any of those messages are produced.

At this point, ensure that your automation product has been updated to trap and handle these messages, and run tests to make sure that the messages are being handled as you expect them to be.

## 5.1.5 Grant RACF access to the SMF log streams

Before you switch to SMF logstream mode, ensure that the data in the log streams is protected from unauthorized accesses, just as you protect the SMF sequential data sets today.

The Manage\_SMF spreadsheet contains a list of all your SMF log streams and the record types contained in each. It also contains a list of the user IDs that process those record types today. Using this information, you are ready to define the access lists for each log stream to your security product.

First, define a generic profile to cover all the SMF log streams. This generic profile will be overridden by specific ones that you define for each log stream, but defining this generic one ensures that if you define additional SMF log streams in the future and forget to set up profiles for those log streams, that they will be protected. To define the generic profile, you can use a command like:

```
RDEFINE LOGSTRM IFASMF.** UACC(NONE)
```

Then define similar profiles for each SMF log stream that you have defined. So, for example, if you have a logstream called IFASMF.PRODPLEX.PERF, set up a profile like this:

```
RDEFINE LOGSTRM IFASMF.PRODPLEX.PERF UACC(NONE)
```

### SMF address space

The user ID associated with the SMF address space needs UPDATE access to all log streams to be able to successfully write the SMF data to the log stream. It is common to define that user ID as TRUSTED so that it can access any required resource. If you do not do that, then you must add the SMF user ID to the access list of every SMF log stream (including the generic one) using a command like:

```
PERMIT IFASMF.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smfas_userid)
```

Remember that you need to do this for each of your SMF log streams.

### Granting access for users of SMF data

At this point in your migration, we assume that you will be writing your SMF data to log streams, then quickly moving the data to the existing sequential data sets. In this scenario, the users of SMF data are unlikely to be accessing the SMF log streams. Rather, they will continue to access the data from the sequential data sets as they do today.

However, if you move to approach 3 or approach 4 in the future, you will need to grant users READ access to the log streams that contain the data that they use.

For now, however, the only user IDs, other than the SMF user ID, that are likely to access the log streams directly are the user ID associated with the archive process and the system programmers working on the migration project.

Depending on what the job is attempting to do, different access levels are required.

If the job is using IFASMF DL with OPTIONS(DUMP) (that is, it is only reading SMF records from the log stream), then read access is required. This is defined using statements like:

```
PERMIT IFASMF.stream1 CLASS (LOGSTRM) ACCESS(READ) ID(ifasmfdl_user1)
PERMIT IFASMF.stream2 CLASS (LOGSTRM) ACCESS(READ) ID(ifasmfdl_user4)
```

If the job is using OPTIONS(ARCHIVE) or OPTIONS(DELETE) (that is, it is going to delete log blocks from the log stream), then update access is required. This is defined using statements like these:

```
PERMIT IFASMF.stream1 CLASS (LOGSTRM) ACCESS(UPDATE) ID(ifasmfdl_arch1userid)
PERMIT IFASMF.stream2 CLASS (LOGSTRM) ACCESS(UPDATE) ID(ifasmfdl_arch2userid)
```

These security permissions apply regardless of whether the user is using the IFASMF DL program or some other program. For example, if the user is using the IFASEXIT program with the LOGR subsystem interface, read access is required for that user to be able to retrieve records from the log stream.

### Permit system programmer access to SMF dump extract

If you get into a situation in which System Logger has failed and you are unable to get SMF to reconnect to the log streams, take a dump of the SMF address space and its associated data spaces *before* you switch back to dataset mode. You can then use the IPCS **SMFDATA** command to extract the unwritten SMF records from the dump into a special log stream called IFASMF.DUMP00. The user who will be responsible for this processing needs update access to that log stream:

```
PERMIT IFASMF.DUMP00 CLASS (LOGSTRM) ACCESS(UPDATE) ID(sysprog)
```

More details on the steps involved in this process are available in “Using IPCS to extract SMF records from a dump” on page 129. This task is generally done by a system programmer.

### Controlling access to offload data sets

As discussed in 4.2.7, “Operator education” on page 74, offload data sets should never be deleted by any mechanism other than when using a DELETE LOGSTREAM command in an IXCMIAPU job, or indirectly by using OPTIONS(DELETE) or (ARCHIVE) in an IFASMF DL job. To reduce the chance of an offload data set from accidentally being deleted, define security profiles to cover the SMF log stream offload data sets (IXGLOGR.IFASMF.\*, for example), and only provide a limited set of users with ALTER access to those profiles.

## 5.1.6 Adjusting the LOGR couple data set

“Review System Logger setup” on page 68 discussed the impact that migrating to SMF logstream mode can have on the utilization of your LOGR CDS. It also showed how to determine the current utilization of that data set and provided a sample JCL to create a new set of CDSs.

If you need to adjust the attributes of the LOGR CDS, now define a new set of CDSs (including one or more spares), and roll them into production with the following command sequence:

```
SETXCF COUPLE,PSWITCH,TYPE=LOGR
SETXCF COUPLE,ACOUPL= new_primary_LOGR_CDS_name,TYPE=LOGR
SETXCF COUPLE,PSWITCH,TYPE=LOGR
SETXCF COUPLE,ACOUPL= new_alternate_LOGR_CDS_name,TYPE=LOGR
```

Remember to add the Logger messages identified in “System Logger messages” on page 76 to your automation product.

## 5.1.7 Defining the SMF Coupling Facility structures

This task is only required if you define log streams residing in a Coupling Facility.

In “Creating the CFRM structure definition statements” on page 59, you created the CFRM definition statements for the structures that you will use to hold the SMF log streams. Now copy those statements into the source for your CFRM policy and run the job to update the CFRM policy.

After the new policy is defined, you must activate it using this command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=new_policy_name
```

## 5.1.8 Defining the log streams

After you have defined the SMF CF structures to the CFRM policy, you are ready to define the log streams that will use those structures in the LOGR policy.

Remember that the LOGR policy is managed differently from CFRM policies. Every time that you submit a new CFRM policy, that is a complete replacement for the existing policy, so any structures that are already defined in your CFRM policy must be included in the job that defines the SMF structures. However, the LOGR policy is updated incrementally. So, when you run a job to define new log streams, you only define those log streams.

### Defining the structures in the System Logger policy

This task is only required if you define log streams residing in a Couple Facility.

In “Creating log stream definition statements” on page 61, you created the statements to define the SMF log streams to System Logger. In those statements you had DEFINE STRUCTURE statements to define any CF structures that the SMF log streams will reside in. Now update the LOGR policy to add those structures, using JCL similar to that shown in Example 5-1.

*Example 5-1 Defining the SMF structures to System Logger*

---

```
//LOGDEF EXEC PGM=IXCMIAPU,REGION=1M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)

DEFINE STRUCTURE NAME(IFASMF_PERF)
          LOGSNUM(4)
          MAXBUFSIZE(65532)
```

```

DEFINE STRUCTURE NAME(IFASMF_CICS)
    LOGSNUM(4)
    MAXBUFSIZE(65532)

DEFINE STRUCTURE NAME(IFASMF_DB2)
    LOGSNUM(4)
    MAXBUFSIZE(65532)

DEFINE STRUCTURE NAME(IFASMF_SEC)
    LOGSNUM(4)
    MAXBUFSIZE(65532)
...
...

```

---

Note that it is necessary to define the structures in the LOGR policy before you issue the DEFINE LOGSTREAM statements that will refer to those structures.

### Defining the log streams in the System Logger policy

The other set of statements that you set up during “Creating log stream definition statements” on page 61 are the ones that define the log streams to the LOGR policy. Now copy those statements into a job like the example in Example 5-2 and run the job.

*Example 5-2 How to define CF structure based and a DASDONLY log stream*

---

```

//LOGDEF    EXEC PGM=IXCMIAPU,REGION=4M
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
DATA TYPE(LOGR) REPORT(YES)
DEFINE LOGSTREAM
    DASDONLY(NO)
    NAME(IFASMF.PRODPLEX.PERF)
    STRUCTNAME(IFASMF_PERF)
    LS_DATACLAS(LOGR24K)
    LS_SIZE(100000)
    STG_DATACLAS(LOGR4K)
    STG_DUPLEX(YES)
    DUPLEXMODE(COND)
    HIGHOFFLOAD(60)
    LOWOFFLOAD(0)
    RETPD(0)
    HLQ(IXGLOGR)
    MODEL(NO)
...
...

```

---

When the job completes, issue a **D LOGGER,L,LSN=IFASMF.\*** command to ensure that the log stream definitions are in line with what you expected.

## 5.1.9 Install and activate the SMF IEFU29L exit

“SMF exits” on page 67 discussed the function of the IEFU29 and IEFU29L exits and when they are invoked. If you determined that you want to use the IEFU29L exit when in logstream

mode, install the exit on your systems at this time. Instructions for how to customize and install the sample exit are provided in Appendix E, “Additional material” on page 135.

### 5.1.10 Prepare the SMF IFASMF DL jobs

In “Submitting archive jobs” on page 92 you looked at the options for submitting the regular IFASMF DL ARCHIVE processes and set up the associated JCL. At this point, place the JCL into the appropriate library and update either your automation or batch scheduling product (depending on which option you selected) to submit those processes at the desired intervals, remembering that DASDONLY log streams can only be accessed from the system that is using that log stream.

It is essential to remember that IFASMF DL DELETE and ARCHIVE operate based on the log block timestamps, compared to DUMP, which operates on the time stamps for the SMF records in each log block. Also, ARCHIVE and DELETE always operate from the beginning of a log stream, whereas a DUMP will start at whatever time and date you specify. This means that doing a DUMP followed by a DELETE is *not* the same as doing an ARCHIVE.

## 5.2 Cutover

Once all tasks described in 5.1, “Preparation” on page 100, have been completed, SMF logstream mode can be activated.

### 5.2.1 Enable SMF logstream mode

The final step in the migration to SMF logstream mode is to switch to the SMFPRMxx members that you will be using in logstream mode.

In 4.2.5, “Preparing SMFPRMxx members” on page 64, we discussed the changes that need to be made to the SMFPRMxx member to exploit and get the best from logstream mode. We also discussed the options, in terms of whether you use a single SMFPRMxx member and switch recording modes using the SETSMF command or maintain two SMFPRMxx members, one specifying RECORDING(LOGSTREAM) and the other specifying RECORDING(DATASET). You should have created the new member (or members) that you will be using at that time.

Now switch to the new SMFPRMxx member that you will be using. If you plan to use the same SMFPRMxx for multiple systems, do not forget to make any changes that might be required in your IEASYMxx member. Specifically, be sure that if the system is IPLed, that it will come back up in the same SMF recording mode that it was in prior to the IPL.

Also, if your strategy is to use a single SMFPRMxx member and use the SETSMF **RECORDING(LOGSTREAM)** command to switch to logstream mode, remember that this currently means that you must specify PROMPT(xxx) in the SMFPRMxx member, and that this will result in an operator prompt during the IPL process. If adding an operator prompt during IPL is not acceptable, then you have no choice other than to use two SMFPRMxx members and use the SET SMF=xx command to switch recording modes.

Starting with z/OS 1.12, the PROMPT option can be used in SMFPRMxx together with the new auto reply function. This allows the system to automatically reply to the SMF operator prompts and might make the use of a single SMFPRMxx member more acceptable. If you

want to exploit this capability, ensure that the Parmlib member used by auto reply contains the following lines:

```
MSGID(IEE357A) REPLY('U') DELAY(1S)
MSGID(IEE956A) REPLY('U') DELAY(1S)
```

This specifies that all prompts resulting from specifying PROMPT(xxx) in your SMFPRMxx member will be automatically answered. Note that this would effectively make it impossible to change the SMF parameters as part of the response to the IEE357A message. However, because the use of PROMPT(xxx) enables the use of the SETSMF command, you can subsequently use that command to make any changes that you want to make. You can also specify a larger DELAY time. However, that has the effect of delaying the IPL every time that you IPL and do not want to change any SMF parameters.

### Verifying successful switch

When you activate logstream mode, monitor the console messages to ensure that no errors were encountered. Also, it is wise to issue a **D SMF,0** command to ensure that the system is using the correct member, that all the log streams are defined as you expected, and that any other changes you make are included in the display.

When you are sure that the system is using the correct member, issue a **D SMF** command. This results in a response similar to that shown in Example 5-3.

*Example 5-3 Displaying log streams usage*

---

```
D SMF
IFA714I 18.04.47 SMF STATUS 853
      LOGSTREAM NAME          BUFFERS      STATUS
      A-IFASMFB.#@#PLEX.CICS    33211      CONNECTED
      A-IFASMFB.#@#PLEX.DB2     14924      CONNECTED
      A-IFASMFB.#@#PLEX.PERF    12521      CONNECTED
      A-IFASMFB.#@#PLEX.DFLT    72548      CONNECTED
```

---

In the response, check to ensure that all the log streams that you expect to see are contained in the list. Also, check that each has a status of CONNECTED. If any of the log streams have a 0 in the BUFFERS column, issue the command a few more times. If it remains at 0, investigate to see why no SMF records are being written to that log stream. Also issue that command a few times over the next few minutes. You normally expect to see that the value in the BUFFERS column (which is the number of bytes currently residing in the data space buffers for that log stream) for each log stream moves up and down as data is built up in the buffer and then moved to the log stream.

## 5.2.2 Verifying successful archive processing

After a certain interval, the process that you are using to archive SMF records from the log streams kicks in. Take note in advance of when you expect the next archive process for each log stream to start, and monitor to ensure that it does start as planned. When it completes, check the output from the job to ensure that it completed successfully. Also check the output data set to ensure that it contains all the SMF records that you expect it to, and specifically that it contains records from the period after the switch.



## 5.3 Post-migration activities

It is important that you closely monitor your SMF logstream mode implementation, particularly in the days immediately after the migration. When running in dataset mode, it is possible to observe the rate of SMF record creation based on the rate at which you fill your MANx data sets. You also have the benefit of years of experience to help you size the SYS1.MAN data sets such that you are capable of offloading SMF data during your peak periods without losing data.

When you move to SMF logstream mode, you also need to monitor SMF activity. However, you use a different set of tools to provide that information. The items to monitor are:

- ▶ SMF Type 88 records for instances of the SMF structures or staging data sets filling
- ▶ Space usage and free space availability in the storage group used for the offload data sets
- ▶ The size and number of offload data sets
- ▶ The use of DSEXTENTS in the LOGR CDS
- ▶ Response times and numbers of requests for the SMF-related CF structures

### 5.3.1 Monitoring Logger activity

To monitor the activity of your log streams, extract the SMF type 88 records and produce a System Logger report. A sample program is delivered in SYS1.SAMPLIB member IXGRPT1J. You can use this to format your SMF type 88 records into helpful reports. Figure 5-1 provides a sample report, with emphasis on one of our log streams. For a description of the meaning of the various fields in the report, refer to the IBM Redbooks publication *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

--LOGSTREAM NAME-----	STRUCTURE NAME--	BYT WRITTN	BYT WRITTN	BYT WRITTN	#WRITES INVOKED	---# WRITES COMPLETED-----							AVERAGE
		BY USERS IXGWITES	TO INTERIM STORAGE	TO DASD		TYPE1	TYPE2	TYPE3	BUFFER SIZE				
		BYT DELETD INTERIM ST W/O DASD	# DELETES W/O DASD WRITE	BYT DELETD INTERIM ST W/DASD		# DELETES W/ WRITE	-----EVENT-----						
						OFF- LOAD	DASD SHFT	STRC FULL	NTRY FULL	STG THLD	STG FULL	RE- BLD	
IFASMF.#@3.TYPRMF	IFASMF_TYPRMF	2503312564	2514747904	2502582824	44307	42506		1606		186		56499	
		0	0	2500812704	44253	274	3	8	8	0	0	0	
IFASMF.#@3.TYPSECU	IFASMF_TYPSECU	20548	22016		6	6		0		0		3424	
		0	0		0	0	0	0	0	0	0	0	
IFASMF.#@3.TYP030	IFASMF_TYP030	84702	86016		6	6		0		0		14117	
		0	0		0	0	0	0	0	0	0	0	
IGWTV003.IGWLOG.SYSLOG	LOG_IGWLOG_001	0	0		0	0		0		0		0	
		0	0		0	0	0	0	0	0	0	0	
IGWTV003.IGWSHUNT.SHUNTLOG	LOG_IGWSHUNT_001	0	0		0	0		0		0		0	
		0	0		0	0	0	0	0	0	0	0	
SYSPLEX.OPERLOG	SYSTEM_OPERLOG	305924	382464	220933	670	670		0		0		456	
		0	0	202253	467	3	0	0	0	0	0	0	

Figure 5-1 IXGRPT1 Logger activity report

In the report shown in Figure 5-1 on page 107 you can see that the IFASMF.#@\$3.TYPRMF log stream was a busy one. In particular, in the reports you want to watch for any instances of the structure or staging data set filling. In this example, you can see that we drove this structure to fill up eight times. However, we did not have any instances of the staging data set filling. The fields that are of particular interest are:

► STRC-FULL: (CF-Structure log streams)

This shows the number of times that a structure-full condition was reached.

This field should be zero for best performance and availability. In this example, the structure was being filled faster than the data could be offloaded. If you observe non-zero values in this field, review the HIGHOFFLOAD and LOWOFFLOAD values of this log stream and the size of the structure.

► STG-FULL: (CF-Structure and DASD-only log streams)

This is the number of times that the staging data set filled in the interval.

This field should be zero for best performance and availability. If the field is non-zero, the possible causes are:

- The CF structure is significantly larger than the staging data set, meaning that the staging data set is filling before an offload is being triggered by the structure reaching the HIGHOFFLOAD threshold.
- There is a problem with the offload process, meaning that System Logger could not move the data to the offload data sets quickly enough.
- The staging data set is too small for the volume of data being written to this log stream.
- The HIGHOFFLOAD threshold is too high, meaning that the space between HIGHOFFLOAD and the data set end is too small.

► NTRY-FULL: (CF-Structure log streams)

This is the number of times that an offload was initiated because the number of entries in use in the structure reached 90%.

This field should be zero for best performance and availability. You might see this if you assigned several log stream to the same structure. Different write rates to the log streams in the same structure can cause this condition. For more information about the element and entry ratio, see *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

## 5.3.2 SMF buffer use

Starting with z/OS 1.12, the SMF Type 7 records contain useful information relating to the use of the SMF buffers when SMF is in logstream mode.

These records contain a count of the number of SMF records that were discarded because the SMF buffers were full when the records were passed to SMF. If there are no instances of the SMF buffers filling, then you should not have any Type 7 records. So, a simple check for the existence of Type 7 records is a easy way to determine whether you are experiencing this situation.

If you are experiencing records being discarded because the SMF buffers are full, those events should be accompanied by a IFA713I SMF DATA LOST message, and your automation package should be raising an alert any time that message is issued. Nevertheless, checking for the presence of these records is an easy and worthwhile check. A sample job to format some of the interesting fields in the Type 7 records is contained in member SMFTYP7 in the RBITSO.CNTL data set in Appendix E, "Additional material" on page 135.

### 5.3.3 Space usage in storage group

One of the most important things for efficient operation of SMF logstream mode is to ensure that nothing slows the offload process. One of the most likely causes of offload problems is a lack of space to allocate new offload data sets. Therefore, it is important to monitor the utilization of the storage group that is used for the SMF log stream offload data sets. There are a number of ways to achieve this:

- ▶ You can use ISMF option 6. (Make sure you specify 'ACTIVE' for the CDS name.)
- ▶ You can use DCOLLECT to obtain information for storage group usage.
- ▶ You can (and should) monitor for instances of the IGD17380I message that indicate that the named storage group has exceeded the high allocation threshold.

If you encounter cases in which the storage group is short of available storage, the most important thing is to immediately provide additional storage to make sure that any future allocation requests for new (or recalled) offload data sets will be successful. After that, you can review your log stream definitions to determine why more storage is being consumed than you had predicted.

### 5.3.4 Number and size of offload data sets

The Stream\_Sizing spreadsheet helped you calculate the number of offload data sets to expect for each log stream. Intermittently check that the actual number of offload data sets is in line with the projections in that spreadsheet. Remember that the name of the offload data sets is the log stream name preceded by the HLQ that you defined for the log stream, and with a suffix that indicates the “generation” number. For example, if the log stream name is IFASMF.PRODPLEX.PERF and the HLQ is IXGLOGR, the offload data sets for that log stream are called something like IXGLOGR.IFASMF.PRODPLEX.PERF.A0000009. This makes it easy to use ISPF Option 3.4 to check the number of data sets for each log stream.

At the same time, check the offload data set sizes. Very small sizes are inefficient, might lead to delays during the offload process, and consume large numbers of DSEXTENTS in the LOGR CDS. Aim for offload data sets that are large enough to contain *at least* one offload from interim storage, but no larger than would hold one day's worth of SMF records for that log stream. Remember that the default offload data set size is only two tracks unless you specify a size in the log stream definition or in the data class that is specified on the LS\_DATACLAS keyword when the log stream was defined.

#### Frequency of allocation of new offload data sets

If you want to set up simple automation to help monitor SMF activity, you can use the IXG283I message that is issued every time that a new offload or staging data set is created. The message contains the data set name and the log stream name. Example 5-4 shows an example.

*Example 5-4 Sample IXG283I message text*

---

```
IXG283I OFFLOAD DATASET IXGLOGR.IFASMF.##$#PLEX.PERF.A0000078 936
ALLOCATED NEW FOR LOGSTREAM IFASMF.##$#PLEX.PERF
CISIZE=24K, SIZE=409927680
```

---

Your automation can extract the log stream name and the time that the message was issued and compare that to the time that the message was last issued for the same log stream. If offload data sets are being allocated more frequently than you deem acceptable, automation can raise an alert. If you are creating offload data sets too frequently but the rate of SMF

record creation is normal for your system, then you need to increase the size of your log streams (LS\_SIZE in the log stream definition).

Another check to do based on the IXG283I message, for all log streams, is to verify that the CISIZE for an offload data set is always 24 K (Example 5-4 on page 109). Using a smaller CISIZE can have a significant negative impact on offload performance.

### 5.3.5 LOGR CDS DSEXTENTS

As discussed in “Review System Logger setup” on page 68, each DSEXTENT in the LOGR CDS can contain 168 offload data sets for a given log stream. If all the DSEXTENTS in the LOGR CDS are in use, and a new DSEXTENT is required to allocate a new offload data set, Logger is unable to allocate the offload data set until a DSEXTENT becomes available. Eventually interim storage for the log stream fills, and error code x'85C' will be returned to the connector the next time that it tries to write to the log stream.

Your automation should be monitoring for Logger messages indicating a shortage of DSEXTENTS and raising an alert if such a message is issued.

You can also run an IXCMIAPU job (as shown in “Review System Logger setup” on page 68) to determine the current usage of DSEXTENTS. If the number of available DSEXTENTS is below a comfortable threshold, allocate a new set of LOGR CDSs with larger DSEXTENT values.

### 5.3.6 Performance of SMF-related CF structures

Given the relative performance of Coupling Facilities compared to DASD, it is unlikely that the CF will be a bottleneck to the performance of your SMF logstream environment. That assumes that you are using CFs with dedicated engines as IBM recommends.

Nevertheless, it is prudent to monitor the response time and request rate for your SMF-related structures.

The request rate varies between structures, depending on the volume and size of SMF records being written to the log stream associated with each structure. There is no “good” or “bad” value. The rate varies from installation to installation. It is more important to monitor the trend to see whether there are unexpected increases or decreases in the volume of requests being sent to each structure.

Also monitor the response time trends. However, generally, do not expect the SMF Logger structures to have as short response times as your other structures. Each read or write to a SMF Logger structure contains between 32 KB and 64 KB of data, compared to 4 KB that would be typical for other structures. As a result, you expect to see that most requests to the SMF Logger structures are asynchronous and longer-running than requests to other structures. Once again, monitor for changes in response times compared with what you normally see for those structures.

## 5.4 Gotchas

Because logstream is a new mode of operation for SMF, you might find that some things behave a little differently than you are used to. In this section, we list some of these situations that we encountered during the writing of this book.

## 5.4.1 Difference between ARCHIVE and DUMP

During your testing, you might run an IFASMF DL DUMP against a log stream, extracting a set of records to a sequential data set. If you then use exactly the same JCL, but specifying ARCHIVE instead of DUMP, you might find that you run out of space in the output data set. A possible reason for this is that the DUMP only extracts records for the time range that you specify, whereas the ARCHIVE processes records from the beginning of the log stream, up to the end time and date that you specify. If the log stream contains many records before the start time that you specify, this might result in the ARCHIVE extracting many more records than the DUMP.

Table 5-1 on page 113 contains a sample IFASMF DL DUMP job.

*Example 5-5 Sample IFASMF DL DUMP job*

---

```
//STEP1    EXEC PGM=IFASMF DL
//OUTDD1   DD DSN=KYNEF.LSDEFLT,DISP=(,CATLG),
//          UNIT=SYSDA,
//          SPACE=(CYL,(100,1),RLSE),LRECL=32760,
//          RECFM=VBS,BLKSIZE=0
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LSNAME(IFASMF.MULTSYS.STREAM1,OPTIONS(DUMP))
OUTDD(OUTDD1,TYPE(0:255))
START(0800)
END(1100)
DATE(2010264,2010264)
/*
```

---

Table 5-2 on page 113 shows exactly the same job, the only difference being that this job specifies ARCHIVE instead of DUMP.

*Example 5-6 Sample IFASMF DL ARCHIVE job*

---

```
//STEP1    EXEC PGM=IFASMF DL
//OUTDD1   DD DSN=KYNEF.LSDEFLT,DISP=(,CATLG),
//          UNIT=SYSDA,
//          SPACE=(CYL,(100,1),RLSE),LRECL=32760,
//          RECFM=VBS,BLKSIZE=0
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
LSNAME(IFASMF.MULTSYS.STREAM1,OPTIONS(ARCHIVE))
OUTDD(OUTDD1,TYPE(0:255))
START(0800)
END(1100)
DATE(2010264,2010264)
/*
```

---

When we ran these jobs, the DUMP job ended successfully, but the ARCHIVE job ended with a return code of 8 when it ran out of space in the output data set. The reason was that the log stream contained data starting on 2010.260, so the DUMP only extracted three hours' worth of SMF records, while the ARCHIVE tried to extract over four days' worth of records.

## 5.4.2 Issuing T SMF in SDSF

When you issue a **T SMF** command from SDSF, specifying a member that contains definitions for both data sets and log streams, the immediate response shows just the data sets. This can be a little misleading. However, if you check the syslog, you will see that both data sets and log streams were successfully defined.

## 5.4.3 IFASMFDL support of OPTIONS(ALL)

In the IFASMFDL program, specifying **OPTIONS(ALL)** indicated that you wanted the SMF records to be read from the **SYS1.MAN** data set and for the data set to then be cleared. Conceptually, it was similar to the processing that **OPTIONS(ARCHIVE)** carries out when using IFASMFDL.

While IFASMFDL still supports the **ALL** parameter on the **OPTIONS** statement, the actions that are carried out are the same as if **DUMP** had been specified. So, when using IFASMFDL, **ALL** is effectively the same as **DUMP**, which is a change from when you are running in dataset mode.

## 5.4.4 Missing offload data sets

If you delete offload data sets by any method other than by doing a **DELETE LOGSTREAM**, System Logger will not be aware that the data sets are missing until it attempts to retrieve log blocks from them. How this impacts IFASMFDL depends on which data sets were deleted and which ones the IFASMFDL job needed. It is possible that the job will issue message **IFA819I** and end with return code 4. Or it might end with return code 0 and just return the records that it was able to find.

However, if the data set that was deleted is the one that contains the beginning of the log stream, an IFASMFDL **ARCHIVE** or **DELETE** will fail because it is unable to access the beginning of the log stream (remember that **ARCHIVE** and **DUMP** *always* start at the beginning of the log stream, regardless of any start time or date that you might specify). The only option in that case is to switch SMF recording to an alternative log stream, use IFASMFDL **DUMP** to extract all the information from the remaining offload data sets, and then delete and redefine the log stream.

The best way to avoid any problems like this is to strictly control the list of users who have security authority to delete the offload data sets, and make sure that those people fully understand the implications of deleting these data sets by any means other than a **DELETE LOGSTREAM** command in an **IXCMIAPU** job.

## 5.4.5 IFASMFDL return codes

There are cases where an IFASMFDL step ends successfully, but ends with a return code of 4 instead of 0 (or vice versa). It is wise to run a variety of tests with IFASMFDL, for example, specifying a range of dates that are not available in the log stream, running a job against a **DASDONLY** log stream on a system other than the one that is currently connected to the log stream, or requesting record types that are not in the log stream, and verify that the return code is what you expected.

## 5.4.6 Protecting SMF data in case of System Logger abend

If SMF is running in logstream mode and System Logger abends, SMF will no longer be able to pass its records over to Logger. In that case, you have two options:

- ▶ Restart System Logger and issue a **T SMF=xx** or a **SETSMF RECORDING(LOGSTREAM)** command to get SMF to reconnect to the log stream.
- ▶ Revert to dataset mode.

If you decide to switch back to dataset mode, you must take a dump of the SMF address space (and its associated data spaces) before you issue the **SETSMF RECORDING(DATASET)** command. If you do not do this, any SMF records that were in the log stream buffers will be lost. You can subsequently use IPCS to extract the unwritten SMF records from the dump. The process to achieve this is described in “Using IPCS to extract SMF records from a dump” on page 129.

## 5.4.7 Z EOD and I SMF commands

The action that the system takes in response to a **Z EOD** command depends on a number of things, as summarized in Table 5-1.

Table 5-1 Actions following Z EOD command

Logstream mode	Data sets defined in SMFPRMxx	IEFU29 invoked	IEFU29L invoked
DATASET	Yes	Yes	No
LOGSTREAM	Yes	Yes	No
LOGSTREAM	No	No	No

The action that the system takes in response to an **I SMF** command depends on a number of things, as summarized in Table 5-2.

Table 5-2 Actions following I SMF command

Logstream mode	IEFU29 invoked	IEFU29L invoked
DATASET	Yes	No
LOGSTREAM	No	Yes

## 5.4.8 Importance of sorting SMF records

Generally, SMF records are stored in the SYS1.MAN data sets in the order in which they are created (that is, the records are in ascending sequence based on the SMFxxDTE and SMFxxTME fields). However, it is possible that user SMF records (where the SMFxxDTE and SMFxxTME fields are completed by the program before the record is sent to SMF) might be marginally out of sequence if there is a delay between when those fields are filled in and when the record is sent to SMF.

Additionally, when operating in logstream mode, and specifically with log streams that contain log blocks from multiple systems, it is likely that the timestamps of the full set of SMF records in the log stream are not completely in chronological order. The timestamps of the records within each log block will be in the same sequence in which they were in the SYS1.MAN data

sets. However, there will be log blocks from multiple systems interleaved in the log stream. This is illustrated in the log blocks shown in Figure 5-2.

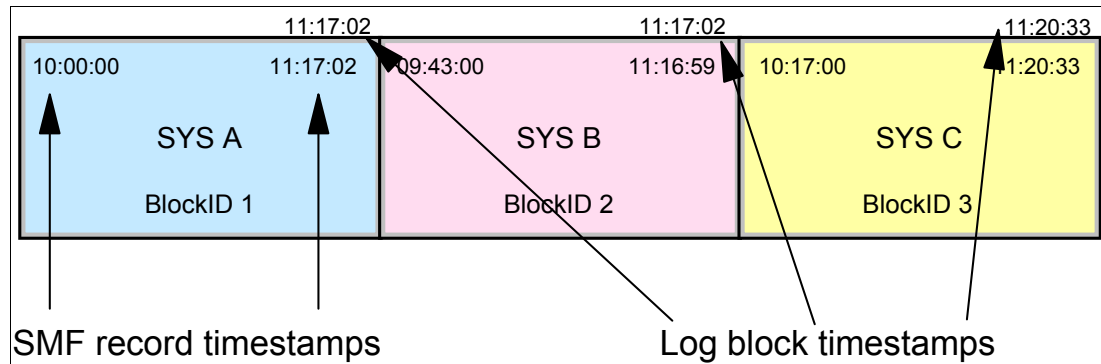


Figure 5-2 Chronological order of SMF records in a multisystem log stream

The log stream contains log blocks from three systems. Within each log block, each subsequent SMF record has the same or a greater SMF timestamp than the previous record. However, when you move from one log block to the next, the SMF timestamps might move back in time.

If the chronological order of the SMF records is important to the program accessing the SMF records, the records should be sorted before use. This applies to both dataset and logstream mode. However, it is more likely to be an issue in logstream mode than in dataset mode and if you are processing data from multiple systems in a single invocation of the program.





# A

## Relative capacity measurements

One of the primary drivers for migrating System Management Facilities (SMF) from dataset mode to logstream mode is to provide more capacity for handling ever-greater numbers of SMF records. To give you an idea of the bandwidth of SMF logstream mode compared with dataset mode, we ran a number of informal measurements.

**Important:** These are *not* a formal IBM benchmark. The hardware configuration that we used for testing was shared with other LPARs, so the results are only indicative of the degree of improvement that you might observe if you migrate from dataset mode to logstream mode in an otherwise-unconstrained environment.

## System Logger performance basics

Because System Logger is a generalized logging facility, designed to handle requests from many types of users, there are a number of things that you can do to influence the performance that System Logger delivers for a given user of its services.

General information about System Logger and how to optimize its performance can be found in *z/OS MVS Setting up a Sysplex*, SA22-7625, and in the IBM Redbooks publication *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

This section provides a summary of the things that you can do to optimize the performance of System Logger when used for SMF log streams.

After covering the basics of System Logger performance, the remainder of this appendix discusses our experiences with a number of measurements that we conducted to understand the possible throughput of SMF in dataset mode compared to logstream mode.

## Optimizing performance for SMF log streams

SMF is, basically, a *funnel* type of System Logger exploiter. That is, it uses System Logger mainly to write. Any retrieval of data will generally be by another program (IFASMF DL, for example). Deletion of data is initiated long after the data is written to the log stream either by IFASMF DL or by System Logger based on the defined retention period for that log stream. This means that, from an SMF perspective, the most important thing is to ensure that data can be written to the log stream as quickly as possible, and that System Logger moves it from interim storage before interim storage fills.

### Interim storage performance

There are two options for interim storage:

- ▶ CF structures
- ▶ Staging data sets

#### **CF**

We address CF first, because that is our preferred option.

The CFs that the SMF log stream structures will reside in should use dedicated engines, in line with IBM recommendations for production sysplexes.

While it is always a good idea to use the fastest CF links possible, in the case of SMF log streams it is especially important to use the highest-bandwidth links that are available to you. Because both the reads and the writes to the SMF log stream structures will be at least 32 KB, and potentially a lot larger, these structure types will benefit more than most structures from access to PSIFB 12X or ICB4 CF links.

If you want to protect the data in the log stream structure from a CF failure, you can either use System Managed Duplexing with the structure, or duplex it to a staging data set. The performance attributes of System Managed Duplexing, and the way it works, mean that it is not ideal for use with SMF log stream structures. You are likely to find that the use of staging data sets will provide at least as good performance, and at a lower cost in terms of used z/OS CPU capacity.

While the amount of storage available in the structures is not strictly a performance question, it is important to ensure that the structure has enough storage to ensure that offloads can complete before the structure fills. See 5.3.1, “Monitoring Logger activity” on page 107, for

information about using SMF Type 88 records to monitor the usage of the structure for each log stream.

### ***Staging data sets***

As you will see in our measurements, CF structures provided significantly more throughput than staging data sets, so staging data sets are not the first choice from a performance perspective.

However, if you have a valid reason for using staging data sets, there are certain things that you can do to optimize performance.

The first suggestion is that you follow all the standard guidance about optimizing DASD performance. Ensure that there are sufficient channels available, that there is no contention on any of the interfaces, and that the device response times are in line with the expectations for that device type.

If doing this still does not deliver acceptable levels of performance, small incremental improvements might be achieved by using SMS data set striping for the staging data sets.

The size of the staging data sets is even more important than the size of the log stream structures. With System Logger structures, pages in the structure are freed up for reuse as soon as the data in each page is moved to an offload data set.

On the other hand, with a staging data set, the space used by pages that have been offloaded is only freed up when the offload completes. This makes it even more important to ensure that the offload can complete in less time than it takes to fill the storage between the HIGHOFFLOAD threshold and when the data set fills.

If you have exhausted all the tuning options for staging data sets, yet there is no way to avoid the use of those data sets, the only remaining option is to break the SMF record types in that log stream across two log streams to reduce the volume of SMF data being written to each log stream.

### **Offload data sets**

The performance and availability of the offload data sets is important from two perspectives:

- ▶ When System Logger performs an offload for a log stream, the performance of the offload data sets should enable the offload to complete as quickly as possible.
- ▶ When retrieving data from an SMF log stream, it is likely that the data will reside in one of the offload data sets, rather than in interim storage. To optimize the performance of any job using SMF data from a log stream, you want retrieval of the data from the offload data sets to be as efficient as possible.

Once again, all the normal guidelines about optimizing DASD apply, just as they do to the staging data sets.

The next thing to ensure is that the offload data sets are set up with a CI size of 24 KB. The use of a *good* CI size has a dramatic effect on how quickly and efficiently System Logger can move data in and out of the offload data sets.

Another thing to consider is possible delays during the offload process. One possible cause is frequent allocation of new data sets during the offload process. This is likely to be caused by specifying an offload data set size that is too small. The SMF Type 88 report that is shown in Figure 5-1 on page 107 contains a field (DASD SHFT) that shows the number of new data sets that were allocated during the interval. If that value is large (more than one per offload), consider altering the log stream definition to increase the LS\_SIZE value.

Another possible cause is delays during recalls of migrated offload data sets. These are not reported directly in the Type 88 records. However, you might notice them if you see cases of log streams intermittently getting full.

If you determine that offloads are not able to keep up with the rate at which data is being written to the log stream, you might try using SMS data set striping for the offload data sets. Based on our experiences, however, the movement of data to the offload data sets from interim storage is much less likely to be an issue than is the performance of the staging data sets.

## Comparative measurements

To understand the scale of improvement that might be achievable when using logstream mode compared to dataset mode, we conducted a number of measurements to compare the throughput and behavior of SMF log streams and SYS1.MAN data sets.

Due to time limitations, we concentrated on measuring the rate at which we could create SMF data in each of these configurations without data being discarded. There are other aspects that you will want to consider, such as:

- ▶ What is the total CPU time used to save each SMF record?

In dataset mode, this is simply the CPU used by the SMF address space.

However, in logstream mode, it is the CPU time used by SMF, and also the CPU time used by System Logger. Consider, however, that System Logger will also be performing work for other requestors, so you cannot simply take all the CPU time used by Logger and charge that to handling of SMF requests.

- ▶ The CPU time used by jobs that subsequently process the SMF records.

In dataset mode, this is simply the CPU time used by the processing program.

However, in logstream mode, there will also be CPU time spent by System Logger as it retrieves the log blocks from the log stream and passes them back to the requesting program. To minimize this time, we suggest using IFASMFDL and making optimal use of its optional keywords to ensure that the only data extracted from the log stream is the data that is actually required by the requesting program.

The configuration that was used for our measurements consisted of:

- ▶ A System z10® model 714
- ▶ IBM ESS, DS8100, and DS8300 DASD
- ▶ One z/OS 1.11 and two z/OS 1.12 systems

The system on which we ran most of our tests had four dedicated CPs and 8 GB of processor storage.

- ▶ Two CF Level 16 Coupling Facilities in the same z10 model 714
  - Each CF LPAR had one dedicated ICF.
  - 12X InfiniBand links were used to connect the z/OS LPARs to the CF LPARs.

We used a program to generate SMF records. The program let us control the number of microseconds between each record that is generated, as well as the sizes of the records. This allowed us to generate a consistent workload for each measurement.

## Methodology

Because the objective of the measurements was to determine the rate at which SMF data could consistently be created without any records being discarded, we needed a way to identify the volume of data being saved in the SMF repository (either SYS1.MAN data set or log streams) per second. By creating SMF records at a rate that ensured that the SMF buffers were always being used, we knew that the repository was being saturated, so by analyzing the usage at that time, we could identify the maximum SMF creation rate that a given configuration could handle.

Setting low BUFUSEWARN values allowed us to see when SMF records were being created faster than they could be moved out of SMF. Because there is no way to display buffer use in dataset mode, setting the BUFUSEWARN threshold to a low value helped us see how many buffers were in use, both in dataset and logstream mode. When the BUFUSEWARN threshold was reached, the IFA785I SMF HAS USED xx% OF AVAILABLE BUFFER SPACE message was issued. For each configuration, we adjusted the rate at which the driver program created SMF records until we had a slow but steady increase in the number of buffers being used.

We needed different mechanisms for dealing with the two SMF recording modes. For dataset mode, we simply had to take the timestamps of the first and last SMF records in the data set that the SYS1.MAN data set was emptied into and the volume of records in the data set (we were able to get all this information from an IFASMFDP report). Dividing the volume of records by the difference in timestamps gave us the highest rate per second that the medium could handle.

For logstream mode, we needed a different mechanism. Remember that a log stream is logically one long stream of data. For most SMF records, the SMF timestamp represents the time that a record was sent to SMF. However, what we needed to measure was the time at which the SMF record was written from the SMF buffers to the log stream. Fortunately, the log block timestamp represents the time that the log block was successfully moved from the SMF buffer to the log stream. Recall that IFASMFDP ARCHIVE processing is based on the log block timestamp. So by doing an ARCHIVE, we could get all the SMF records from the beginning of the log stream until the end time that we specified on the ARCHIVE. This gave us an amount of time. Then we got the volume of SMF data (from the IFASMFDP report). Dividing the volume of data by the time interval gave us the rate per second.

## Measurement

The scenarios that we measured were:

- ▶ SMF writing to a SYS1.MAN data set on an IBM 2105
- ▶ SMF writing to a SYS1.MAN data set on an IBM 2107
- ▶ SMF writing to a single DASDONLY log stream on 2107
- ▶ SMF writing to a single CF-only log stream
- ▶ SMF writing to two CF log streams
- ▶ SMF writing to a CF log stream that was duplexed to a staging data set
- ▶ SMF writing to a CF log stream that was duplexed to a staging data set that was striped across four volumes
- ▶ SMF writing to a CF log stream that was duplexed to a staging data set that was striped across four volumes and using offload data sets that were also striped

The two measurements using the SYS1.MAN data sets were intended to provide a base against which the other measurements could be compared. Also, by doing a measurement with the SYS1.MAN data sets on IBM 2105 DASD and another measurement with them on

IBM 2107 DASD, we were able to see to what extent different DASD technology can improve SMF throughput.

The next test (writing all SMF record types to a single DASDONLY log stream) was intended to provide as close as we could get to a like-for-like comparison between SMF writing to its own Virtual Storage Access Method (VSAM) data sets, and SMF using System Logger to write to a staging data set. We do not expect customers to actually use this configuration in production, but it does provide an interesting view of the benefit that the use of System Logger can provide compared to ordinary VSAM data sets.

We then took a measurement of a single CF log stream. This allowed us to compare the relative throughput of a staging data set with a CF structure.

The next measurement was of two log streams in the CF, and was intended to investigate the scalability that the use of multiple log streams might be able to provide.

The last set of measurements was intended to show the relative throughput of a CF log stream that contained critical SMF records, and how tuning the staging and offload data sets would affect the throughput.

## Measurement results

Figure A-1 shows the different throughputs we achieved for each of the different configurations.

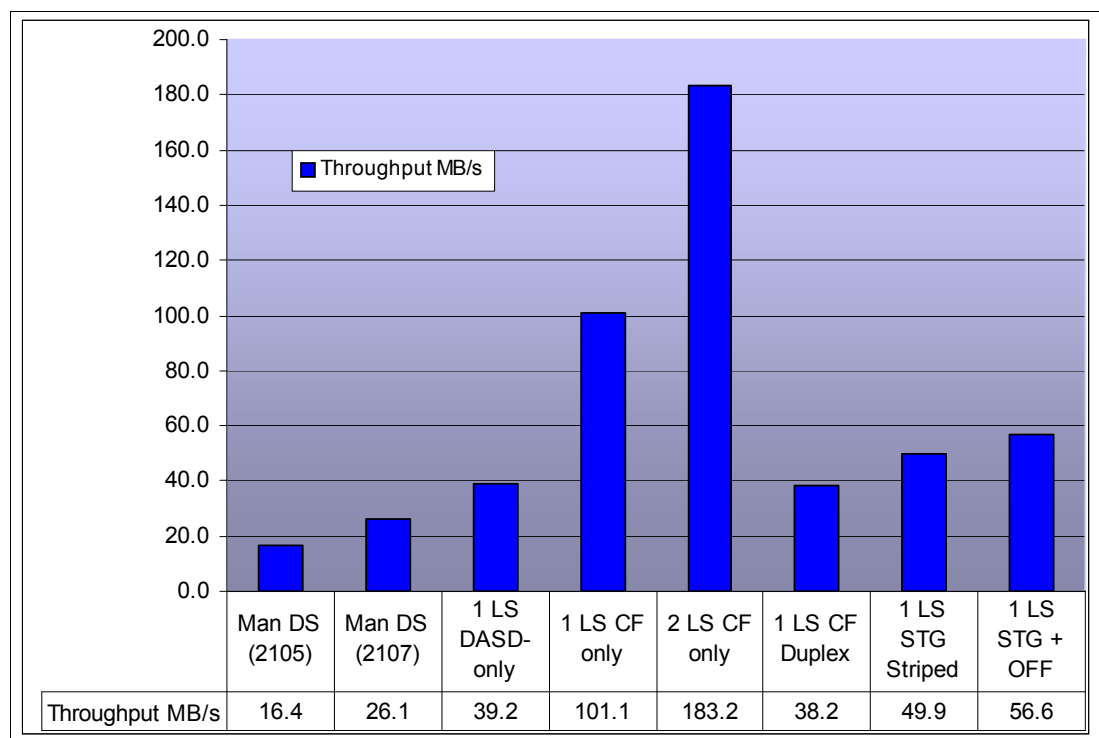


Figure A-1 Measurement results

As you can see, moving the SYS1.MAN data sets from the IBM 2105 DASD to the IBM 2107 DASD delivered a 59% increase in throughput. This is a worthwhile improvement, and shows that if you are concerned about the rate at which you are creating SMF data, you should place your SYS1.MAN data sets on the fastest device available. Nevertheless, it is just 25% of the throughput that we achieved with just one log stream when placed in a CF.

When SMF was changed from dataset mode to logstream mode, writing to a single DASDONLY log stream, the throughput that SMF was able to sustain increased by 50%. Note that the staging data set was on the same type of device as the SYS1.MAN data set (IBM 2107), yet System Logger's more efficient use of VSAM resulted in increasing SMF throughput capability by a half.

The next measurement shows how writing to a CF structure, instead of staging data sets, can dramatically increase throughput. In this case, there was a 257% increase in the volume of SMF data that could be processed without any data being discarded. Again, we do not think that installations will run with just a single log stream, but this measurement is useful because it highlights the relative performance of a CF-only log stream compared to the throughput that can be achieved for log streams that require the use of a staging data set.

The next measurement consisted of splitting the SMF data over two log streams to see how scalable SMF use of log streams would be. In this measurement, we observed an increase of over 81% in the volume of SMF data that could be handled, simply by moving from one log stream to two.

However, it is likely that nearly every installation will have some SMF record types that are deemed critical to the business, and therefore *must* be duplexed to staging data sets. Recall that we suggested the use of CF log streams even for these record types, so we wanted to see how duplexing the log streams to staging data sets would impact the performance. As you would expect, the throughput we observed was close to that observed with a DASDONLY log stream. Because every write to the log stream in the CF must be followed by a synchronous write to a staging data set, the throughput of any log stream that uses staging data sets will be closer to the throughput of a DASDONLY log stream than that of a CF-only log stream. In reality, we expect that customers will have a combination of CF-only log streams and a smaller number of CF+staging log streams.

The final two measurements were intended to determine whether the use of striping would provide any noticeable benefit for log streams that are using staging data sets. The first measurement involved striping the staging data set. This resulted in a throughput increase of 30%. In an effort to speed up the movement of log blocks out of the staging data sets during offload, we then tried striping the offload data sets as well. This resulted in a further increase of 13% in the volume of SMF data that could be handled by that log stream.







## Important command and message changes

When you migrate from System Management Facilities (SMF) dataset mode to logstream mode, some commands present different output, some have a different effect, and there are some new ones that you might not have used previously. This appendix provides information about these commands.

There are also new messages specifically related to SMF logstream mode (these are generally in the IFA7nn and IFA8nn range). These messages should be understood so that any problems can be addressed in a timely way. Some of them should also be added to your automation product to ensure that they are immediately brought to the attention of the appropriate staff.

## SMF commands

While the process of emptying your SYS1.MAN data sets is almost certainly automated, it is possible that unexpected situations might require operator intervention. Because SMF has been around for so long, most operators will be familiar with a set of SMF commands that they can use to understand what is happening and to control SMF behavior. This section describes those commands and how they might change when running in logstream mode.

### D SMF

There are a number of variations of the **D SMF** command. The most basic one, **D SMF**, provides information about the recording medium currently being used to hold SMF records.

Example B-1 shows an example of the response when in dataset mode.

*Example B-1 D SMF command in dataset mode*

---

```
D SMF
IEE974I 13.03.50 SMF DATA SETS 497
      NAME                VOLSER SIZE(BLKS) %FULL  STATUS
P-SYS1.#@$.MANC          #@$#SA   90000    5  ACTIVE
S-SYS1.#@$.MAND          #@$#SA   90000   100  DUMP REQUIRED
```

---

This shows the status of each SYS1.MAN data set, the number of data sets defined to SMF, the size of each one, and how full it is. Both the status and the %FULL are particularly important. If there is a problem with the process that empties the SYS1.MAN data sets, you will likely see multiple data sets with a status of DUMP REQUIRED.

Example B-2 shows the equivalent response when D SMF is issued and SMF is in logstream mode. In logstream mode, the type of information that is provided is somewhat different. For one thing, a SYS1.MAN data set has a finite size, so it is logical to report on how full they are. However, log streams do not have a finite size, so it is not possible to talk about how “full” they are. Instead, the **D SMF** command shows you which log streams are defined and how many bytes of data are currently sitting in the SMF buffers waiting to be moved to each log stream.

*Example B-2 D SMF response when in logstream mode*

---

```
D SMF
IFA714I 13.31.02 SMF STATUS 984
      LOGSTREAM NAME                BUFFERS      STATUS
A-IFASMF.#@$2.TYPDFLT              35914    CONNECTED
A-IFASMF.#@$2.TYPDB2                  0    CONNECTED
A-IFASMF.#@$2.TYPCICS                  0    CONNECTED
A-IFASMF.#@$2.TYPMQ                   0    CONNECTED
A-IFASMF.#@$2.TYPTCP                   0    CONNECTED
A-IFASMF.#@$2.TYPSECU                1963    CONNECTED
A-IFASMF.#@$2.TYPRMF                  0    CONNECTED
A-IFASMF.#@$2.TYP030                  0    CONNECTED
```

---

You can see in this example that eight log streams are defined to this system. Two of them currently have a number of records that are waiting either for the MAXDORM period to expire or for a full block’s worth of data to be assembled. Despite the fact that the heading says BUFFERS, that column contains the number of bytes that are currently in the SMF buffers for that log stream. As our system was set up to use a blocksize of 64 K, you can see that for both of these log streams, SMF is still waiting for more records to arrive so that it can fill the block and send it to Logger.

This command is a quick and easy way to see which log streams are receiving the largest number of SMF records by observing the buffer values as you issue multiple displays.

The maximum amount of buffer space for each log stream is 2 GB, or 2147483648 bytes. Prior to z/OS 1.12, there was no advance warning if the log stream buffers started to fill, so the **D SMF** command was the only way to determine how much SMF data was sitting in the buffers. If you issue this command a number of times and discover a larger than normal value for one or more log streams, contact your system programmer to investigate whether there are problems that are stopping System Logger from being able to successfully offload the log streams to the offload data sets.

Another useful command is the **D SMF,0** command. This command shows the SMF options that are currently in effect. The command provides the same type of output, regardless of whether SMF is in dataset or logstream mode. However, if you are making changes to SMF, use this command to verify that your changes took effect.

## I SMF

When running in dataset mode, the **I SMF** command is used to switch from one SYS1.MAN data set to the next available one. As part of this processing, the command also causes SMF to flush all its buffers out to the SYS1.MAN data sets. This can be useful if you want to gather all the SMF records up to a particular time, and want to be sure that all records before that time have been moved out to the data sets. The last thing that command does is to drive the IEFU29 exit. This is typically used to kick off a started task or a batch job that will move all the data out of the just-switched-from SYS1.MAN data set and make it available for use again.

When running in logstream mode, there is no concept of switching between data sets (remember that the log stream is just one long stream of data). However, the **I SMF** command still carries out some of the same actions as when SMF is in dataset mode. When the command is issued, SMF closes any partially built blocks that it has and sends them to System Logger. When that has completed, it drives the IEFU29L exit. This is similar to the IEFU29 exit, but is for when SMF is in logstream mode. So, for example, you could have automation issue an **I SMF** command every hour. This would force all buffers to be written to the log streams, and then the IEFU29L exit could kick off a job or started task that would move the last hour's worth of log records from each log stream to a sequential data set.

## Z EOD

It is still important to issue the **Z EOD** command during system shutdown. Because there is more space for SMF buffers in logstream mode (up to 2 GB per log stream) than in dataset mode (maximum of 1 GB of buffer space), you could potentially lose more SMF data in logstream mode if you shut the system down without giving SMF a chance to empty all its buffers into the corresponding log streams.

## SET SMF and SETSMF commands

There are two ways to change the options in use by SMF:

- ▶ Switch to a new SMFPRMxx member. This is achieved by using the **SET SMF** command (for example, **SET SMF=01**). In this case, SMF switches to a new member and sets all its options to match the specifications in that new member.
- ▶ Use the SETSMF command to specify a particular option that you want to change, for example, **SETSMF RECORDING(LOGSTREAM)**. In this case, only a single option is changed.

To be able to use the SETSMF command, the SMFPRMxx member must specify PROMPT(xxx). However, specifying this parameter means that the operator will be prompted every time that the system is IPLed or the **SET SMF** command is used to switch to a new SMFPRMxx member.

Neither method is better than the other. Some customers prefer to implement changes by updating the SMFPRMxx member and then switching to that member. Others prefer to use the SETSMF command to implement changes. Follow the conventions used by your site.

## SMF messages

One of the most serious conditions for SMF is losing the ability to externalize its records. If SMF is in dataset mode and no SYS1.MAN data set is available for use, SMF starts to build up the records in its buffers. In this case, you receive a set of messages informing you that SMF has used more than x% of its buffers (where x is specified by the installation on the MAXBUFUSEWARN keyword in SMFPRMxx). Example B-3 shows an example of this situation.

*Example B-3 SMF messages when all SMF data sets are full*

---

```

IEE974I 13.03.50 SMF DATA SETS 497
      NAME                VOLSER SIZE(BLKS) %FULL  STATUS
      P-SYS1.#@$A.MANC    #@$#SA    90000    5  ACTIVE
      S-SYS1.#@$A.MAND    #@$#SA    90000   100  DUMP REQUIRED

I SMF
START SMFCLR,DSNAME=SYS1.#@$A.MANC
IEFU29 has issued command 'START SMFCLR,DSNAME=SYS1.#@$A.MANC
'

IEF196I IEFU29 has issued command 'START SMFCLR,DSNAME=SYS1.#@$A.MANC
IEF196I
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 928
      TO START SMFCLR WITH JOBNAME SMFCLR.
$HASP100 SMFCLR  ON STCINRDR
IEE366I NO SMF DATA SETS AVAILABLE--DATA BEING BUFFERED TIME=13.05.20
IEF695I START SMFCLR  WITH JOBNAME SMFCLR  IS ASSIGNED TO USER STC
, GROUP SYS1
$HASP373 SMFCLR  STARTED
*IEE986E SMF HAS USED    25% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    26% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    27% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    28% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    29% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    30% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    31% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED    32% OF AVAILABLE BUFFER SPACE

```

---

Prior to z/OS 1.12, it was not possible to specify a MAXBUFUSEWARN value for SMF log streams. However, starting with z/OS 1.12, you can specify a MAXBUFUSEWARN value for each log stream. It is important to monitor for, and react to, the messages that are issued if the MAXBUFUSEWARN threshold is exceeded (Example B-4).

*Example B-4 Buffer usage messages in logstream mode*

---

```

$HASP373 SMFDRV01 STARTED - INIT 1    - CLASS A - SYS #@$3
*IFA785E SMF HAS USED 26% OF AVAILABLE BUFFER SPACE FOR 744
      IFASMF.#@$3.TEST
*IFA785E SMF HAS USED 29% OF AVAILABLE BUFFER SPACE FOR 745
      IFASMF.#@$3.TEST
*IFA785E SMF HAS USED 32% OF AVAILABLE BUFFER SPACE FOR 746
      IFASMF.#@$3.TEST
*IFA785E SMF HAS USED 35% OF AVAILABLE BUFFER SPACE FOR 747

```

---

```
IFASMF.#@$3.TEST
$HASP395 SMFDRV01 ENDED
```

---

The old idea of having operators monitoring an MVS console is simply no longer possible. The rate at which messages are produced on any busy z/OS system is impossible for any human to keep up with. Therefore, critical messages (such as the IFA785E and the related IFA786W messages) that you want the operators to be aware of must be defined to the automation product so that it can forward them to the operator. A list of suggested SMF messages to monitor for is included in “SMF messages” on page 75.

## System Logger commands

When SMF is running in logstream mode, there are two system components involved in the writing of each SMF record:

- ▶ SMF
- ▶ System Logger

Therefore, to monitor the successful operation of SMF, there are Logger commands that you must be familiar with.

### D LOGGER,CONN,JOB=SMF

To determine Logger’s view of its connection with SMF, you can use the **D LOGGER,CONN** command. This command provides an overview of which log streams are currently connected to SMF on the system where the command is issued. Example B-5 shows a sample response.

*Example B-5 D LOGGER,CONN command*

---

```
D LOGGER,CONN,JOB=SMF
IXG601I 19.55.24  LOGGER DISPLAY 962
CONNECTION INFORMATION BY JOBNAME FOR SYSTEM #@$2
JOBNAME: SMF      ASID: 001B
LOGSTREAM          STRUCTURE      #CONN  STATUS
-----
IFASMF.#@$#PLEX.TYPRMF  IFASMF_TYPRMF  000001 IN USE

IFASMF.STRIPE.TYPDFLT   *DASDONLY*    000001 IN USE

NUMBER OF LOGSTREAMS: 000002
```

---

In this example, you can see that on system #@\$2, SMF is currently connected to two log streams. Both log streams are only being used by one system (#CONN). One log stream is a DASDONLY one, and the other is using a CF structure.

## D LOGGER,CONN,SYSPLEX,LSN=IFASMF.\*

To get a sysplex-wide view of your SMF log streams and where they are connected, the keyword **SYSPLEX** can be used in the **D LOGGER,CONN** command. In this command you can take the advantage of the wildcard support on the **DISPLAY LOGGER** command, remembering that all SMF log stream names must start with **IFASMF.\***. Example B-6 shows an example of the use of the command.

*Example B-6 D LOGGER,CONN,SYSPLEX,LSN command*

---

```
D LOGGER,CONN,SYSPLEX,LSN=IFASMF.*
IXG601I  20.02.18  LOGGER DISPLAY 953
CONNECTION INFORMATION FOR SYSPLEX #@$#PLEX
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----
IFASMF.#@$#PLEX.TYPRMF  IFASMF_TYPRMF  000002  IN USE
  SYSNAME: #@$2
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: #@$A
    DUPLEXING: LOCAL BUFFERS
  GROUP: PRODUCTION
IFASMF.#@$A.TEST        IFASMF_TEST      000001  IN USE
  SYSNAME: #@$A
    DUPLEXING: STAGING DATA SET
  GROUP: PRODUCTION
IFASMF.STRIPE.TYPDFLT    *DASDONLY*      000001  IN USE
  SYSNAME: #@$2
    DUPLEXING: STAGING DATA SET
  GROUP: PRODUCTION

NUMBER OF LOGSTREAMS:  000003
```

---

In this example, you can see that a total of three SMF log streams are in use in the sysplex. The log stream called **IFASMF.#@\$#PLEX.TYPRMF** is being used by two systems, and the other two log streams are each being used by just one system. You can also see where Logger is holding its second copy of the log blocks that are currently in the interim part of the log stream (on the **DUPLEXING:** line).

## D LOGGER,L,LSN=IFASMF.\*

In addition to displaying information about the SMF log streams that SMF is currently connected to, you can also display a list of *all* the SMF log streams that are defined to Logger (Example B-7).

*Example B-7 D LOGGER,L,LSN=IFASMF.\* command*

---

```
D LOGGER,L,LSN=IFASMF.*
IXG601I  20.14.04  LOGGER DISPLAY 961
INVENTORY INFORMATION BY LOGSTREAM
LOGSTREAM          STRUCTURE          #CONN  STATUS
-----
IFASMF.#@$#PLEX.TYPALL  IFASMF_TYPALL  000000  AVAILABLE
IFASMF.#@$#PLEX.TYPRMF  IFASMF_TYPRMF  000002  IN USE
  SYSNAME: #@$2
    DUPLEXING: LOCAL BUFFERS
  SYSNAME: #@$A
    DUPLEXING: LOCAL BUFFERS
  GROUP: PRODUCTION
```

IFASMF.#@\$A.TEST	IFASMF_TEST	000001 IN USE
SYSNAME: #@\$A		
DUPLExING: STAGING DATA SET		
GROUP: PRODUCTION		
IFASMF.#@\$3.TEST	IFASMF_TEST	000000 AVAILABLE

---

In this example, you can see that two SMF log streams, IFASMF.#@\$#PLEX.TYPALL and IFASMF.#@\$3.TEST, are defined, but they are not currently in use (#CONN is 000000). If you detect such a situation, immediately investigate whether this is expected or whether there is an error that has caused SMF to disconnect from some of the log streams.

## Other commands

You can get information about the structures that are being used by CF-only or CF+staging log streams. The **D XCF,STR,STRNAME=i fasmf\_name** command provides information such as whether the structure is currently allocated and, if it is allocated, which CF it resides in, what its current size is, how many systems are connected to the structure, and the current usage of the structure. If you are experiencing availability or performance problems with a CF log stream, the output from this command might be helpful in subsequent investigation of the problem.

## Using IPCS to extract SMF records from a dump

There might be situations in which the only copy of certain SMF records is in a dump. One example might be if you specify NOBUFFS(HALT) and then take a standalone dump if the buffers for a log stream fill. Another example is if you are running in logstream mode and lose connectivity to the SMF log streams and are unable to reconnect. In that case, you must take a dump of the SMF address space and its associated data spaces to capture the SMF records that are in the log stream buffers.

The following example shows how you use IPCS to extract SMF records from a dump of a system that was running in logstream mode. The example assumes that you have previously defined a log stream called IFASMF.DUMP00.

1. Preparation: To make the process of capturing an SMF dump easier, we suggest that you prepare an IEADMCxx Parmlib member in advance. The contents of the member would look like the sample statements in Example B-8.

*Example B-8 Parmlib member to dump SMF*

---

```
TITLE=(SMFAS)
JOBNAME=(*MASTER*,SMF)
DSPNAME=('SMF'.*)
```

---

You can then use the **DUMP PARMLIB=xx** command to take an SVCDUMP on each system that cannot connect to the log stream.

2. After you capture the SMF dump, you can switch back to dataset mode (if that is deemed to be the correct action) using the **SET SMF=xx** command to switch to a member that contains the definitions of the SYS1.MAN data sets.

3. The next step is to extract the SMF records from the dump using IPCS. The easiest way to do this is to run an IPCS batch job (Example B-9). Note that SMF records from all log streams are merged into the IFASMF.DUMP00 log stream.

*Example B-9 IPCS SMFDATA step*

---

```
/** WRITE SMFDATA TO IFASMF.DUMP00 LOGSTREAM *****
//IPCS      EXEC PGM=IKJEFT01,REGION=6M,DYNAMNBR=10
//IPCSDDIR DD DSN=KLAIEY.DDIR,DISP=SHR
//SMFDUMP DD DSN=DUMP.D0805.H12.##$3.##MASTER#.S00003,DISP=SHR
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSPRNT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
IPCS
SMFDATA DDNAME(SMFDUMP)
DROPDUMP DDNAME(SMFDUMP)
END
/*
```

---

Note that, at the time of writing, the **IPCS SMFDATA** command might abend with an S0C4 if the normal SMF log streams have a MAXBUFSIZE greater than 61000. The problem is addressed by FIN APAR OA34045. Until a z/OS release that addresses this problem is released, you can circumvent the problem using a slip trap similar to this:

```
SLIP SET,IF,PVTMOD=(IFASMFDT,0764,0764),JOBNAME=jobname,ACTION=REFBEFOR,REFBEFOR
=(2R,EQ,11000),END
```

Ask IBM support for the current offset, as it might change as a result of applying service.

4. At this point, the SMF records from the dump have been moved to the IFASMF.DUMP00 log stream, so run an IFASMF.DL DUMP to extract the data from the log stream to a data set.





## Considerations for systems without APAR OA34589

Prior to APAR OA34589 (which will only be available for z/OS 1.11 and 1.12), there was a restriction that an IFASMF DL ARCHIVE or DELETE could not delete all the log blocks in a log stream.

If your process is to do ARCHIVES or DELETES on a daily, weekly, or monthly basis, and you specify something like `RELATIVEDATE(BYxxx,2,1)`, so there is an implied end date and time that is in the past, then this restriction should not be an issue. However, in general, if your methodology is to archive the System Management Facilities (SMF) records from the log stream to sequential data sets, then it is wise to run the archive more frequently than once a day in order to minimize the elapsed time for the archive job. Equally, if your methodology is to do regular ARCHIVE or DELETES, and you have a mechanism for dynamically generating DATE and END parameters that will generate an end time that is in the past, then you also should not have an issue. Of course, this is more complex than simply being able to submit the same JCL every time because you will need to build a program or process to generate these DATE and END parameters every time you submit the job.

Specifying `RELATIVEDATE(BYDAY,0,1)` might help matters, because it sets an implicit end date and time that is equal to the start time of the program. If the log stream is a busy one, then it is likely that more log blocks will be written to the log stream before the program reaches the implicit end time, meaning that the archive operation will not empty the entire log stream. However, if the log stream is not very busy, or if there are few log blocks to be processed, then there is the possibility that the ARCHIVE process will reach the implicit end time before any more log blocks have been written to the log stream, meaning that the ARCHIVE would try to delete all the contents of the log stream, and that is not permitted.

If the ARCHIVE or DELETE operation would result in all log blocks being deleted from the log stream, the ARCHIVE operation ends with return code 8, no log blocks are deleted from the log stream, and the only message you receive is:

```
IFA832I INVALID PARAMETER COMBINATION FOR ARCHIVE OPTION
```

One thing that adds to the complexity of handling this situation is that the IFASMF DL program does not know whether the ARCHIVE operation will result in deleting all the log blocks until it

reaches the end point. However, at that point, it has already copied the SMF records to the data set specified on the OUTDD statement. If each ARCHIVE mods on to an existing data set, and you then run the ARCHIVE job again later, you end up with duplicate records in the OUTDD data set.

One way around this is to output to a new data set every time that you run an ARCHIVE. If the ARCHIVE step ends with a return code greater than 4 (meaning that it has not deleted the log blocks from the log stream), a subsequent step can delete the just-created data set.

An alternative is to subsequently (maybe at the end of each day) post-process the SMF sequential data sets with a program that will delete duplicate records. This is something that you might already be doing, because the risk of having duplicate SMF records in the sequential data set is not unique to SMF logstream mode.

Specifying a small MAXDORM value might also help, as this has the effect of forcing SMF to write log blocks more frequently if the log stream has a low write rate. However, if the users of the log stream only write SMF records once per SMF interval (RMF, for example), then MAXDORM does not cause log blocks to be created between those intervals.

The net is that the only way to be sure that you do not try to delete all log blocks from the log stream is to specify an END time on the IFASMF DL ARCHIVE or DELETE job that is at least one SMF interval before the current time.



## Changing log stream attributes

Many log stream attributes can be changed dynamically. Information about this process can be found in the sections titled “Upgrading an existing log stream configuration” and “Updating a log stream’s attributes” in *z/OS MVS Setting up a Sysplex, SA22-7625*. However, a small number of attributes can only be changed by disconnecting all users from the log stream. This section provides a process to implement such a change.

At the time of writing, the keywords that require that all connectors disconnect from the log stream are `DESCRIPTOR`, `GROUP`, and `STRUCTNAME`. Fortunately, it is possible to change these attributes for a System Management Facilities (SMF) log stream without stopping SMF. Because you can dynamically change the log stream that SMF is sending a given record type to by updating and activating the `SMFPRMxx` member, you can cause SMF to disconnect from a log stream without having to stop it or do an IPL.

You do not want to be switching SMF between log streams on a regular basis, but in the small number of cases where disconnecting from a log stream is really necessary, it *is* possible to do so. You can use the following steps to activate new log stream attributes when disconnects are needed. (This process should be done at a quiet time, preferably when no one other than SMF will be accessing the log stream that you want to change.)

1. Assume that the log stream called `IFASMF.SOME` is the log stream that you want to update. The first thing that you need to do is to define a new, temporary, log stream called `IFASMF.SOME.TEMP`, with attributes matching the current log stream.
2. Ensure that whoever has security access to the `IFASMF.SOME` log stream also has access to the `IFASMF.SOME.TEMP` log stream.
3. Set up a new temporary `SMFPRMxx` member called `SMFPRMTMP` (for example). Change the `LSNAME` statement that points to `IFASMF.SOME` to point at `IFASMF.SOME.TEMP` instead.

Rather than setting up a new log stream, you *can* either assign the record types for this log stream either to another `LSNAME` or let the system assign them to the `DEFAULTLSNAME`. However, if you do this, you might need to also make temporary changes to any `ARCHIVE` jobs for those log streams, to ensure that the record types that normally go into `IFASMF.SOME` are handled correctly. You also need to be careful not to mix records that will be archived with records that will be kept in the log stream in the

same log stream. For these reasons, we believe that it is easier to set up a new, temporary log stream specifically to replace IFASMF.SOME during the change.

4. Activate the new SMFPRMTP member using the **SET SMF=TP** command. You *must* do this on *every* system in the sysplex that is using that log stream.

When SMF sees that log stream IFASMF.SOME has been removed, it empties all its buffers for that log stream into the log stream before it disconnects from it.

5. Verify that log stream IFASMF.SOME no longer has any connection to SMF by issuing a **D LOGGER,CONN,SYSPLEX,LSN=IFASMF.SOME** command. If SMF is still connected, wait a little while for it to complete writing to the log stream and then check again.
6. After all the SMFs disconnect (meaning that all the buffers have been written to the log stream), run an IFASMF DL ARCHIVE against the log stream to move all the records to the associated sequential data sets.
7. After the archive job completes, update the IFASMF.SOME log stream attributes using the IXCMIA PU utility (Example D-1). In Example D-1, the log stream is being moved to a new structure (one of the few changes that requires disconnection from the log stream).

---

*Example D-1 Update attributes of an existing log stream*

---

```
//UPDATELG EXEC PGM=IXCMIA PU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(LOGR) REPORT(YES)
UPDATE LOGSTREAM NAME(IFASMF.SOME)
STRUCTNAME(IFASMF_SOMEBIGR)
/*
```

---

8. Issue a **D LOGGER,L,LSN=IFASMF.SOME** command and verify that the structure associated with log stream IFASMF.SOME is the one that you expect.
9. Move back out to the original SMF setup with IFASMF.SOME by issuing a **SET SMF=00** command (assuming that SMFPRM00 is your original configuration).
10. Run an IFASMF DL ARCHIVE job to retrieve any SMF records that were written to IFASMF.SOME.TEMP during the change.
11. Delete the IFASMF.SOME.TEMP log stream.

There are also a small number of log stream attributes that cannot be updated, regardless of the connections to the log stream. The following log stream attributes cannot be changed without deleting and redefining the log stream:

- ▶ HLQ
- ▶ EHLQ
- ▶ LIKE
- ▶ MODEL

Therefore, these attributes need to be planned especially carefully.

In addition to these items, we suggest that you review the IBM Redbooks publication *System z Parallel Sysplex Best Practices*, SG24-7817, for best practice guidelines for System Logger.



## 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 publication web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247919>

Alternatively, you can go to the IBM Redbooks publication website at:

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

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247919.

### Using the web material

The additional web material that accompanies this book consists of two files:

- ▶ SG247919\_Ssheets.zip

This file consists of a folder that contains the Manage\_SMF and Stream\_Sizing spreadsheets.

- ▶ SG247919\_z0Stools.zip

This file consists of a number of MVS data sets. To use these, upload the data sets in BIN format to sequential FB 80 data sets on z/OS. Then issue a TSO RECEIVE command against each of the data sets.

# Spreadsheets

The tools included in the SG247919\_Ssheets zip file are:

- |                          |  |
|--------------------------|--|
| <b>Manage_SMF.xls</b>    | A spreadsheet to act as a repository of information that will help you decide how to manage each System Management Facilities (SMF) record type. |
| <b>Stream_Sizing.xls</b> | A spreadsheet to help size and configure your SMF log streams based on an IFASMFDP output.   |

## Manage\_SMF spreadsheet

The intent of the Manage\_SMF spreadsheet is to help you assemble into a single place all the information that you need to help you determine the best way to manage each SMF record type. You might find that you require additional information, in which case you can modify the spreadsheet accordingly, but this sample will at least get you started.

The meanings of the columns in the spreadsheet are:

► Record type

There should be one row per SMF record type. Because the granularity of direction SMF data to specific log streams is at the record type (rather than subtype) level, it should not be necessary to be more detailed than one row per record type.

► Collected?

Flag to indicate if you are currently collecting this SMF record type.

► SEQ DSNs

Names of data sets that contain this record type (base GDG names or other data sets that contain these SMF records). Get this information from the SMFDSACC program.

► Accessing programs

Name of programs that access those data sets. Get this from the SMFDSACC job.

► Accessing jobs

Names of jobs (or started tasks) that access the data sets containing these SMF records. You should get this from the SMFDSACC job. This information might help you find the JCL source if JCL changes are required.

► Accessing Userids

User IDs that accessed the data sets. Get this from the SMFDSACC job.

► Retention period

How many days does this record type need to be kept for?

This information, together with the MB/day value, helps calculate how large the log stream offload data sets need to be.

The SMFDSACC job might help by providing the retention period or the number of generations for generation data groups.

► Processed at sysplex or system level?

This is an indicator of whether this record type is processed at the sysplex or system level. If at the sysplex level, this record must go in a CF log stream.

- ▶ Lost records acceptable?

This is an indication of whether it would be acceptable to lose a small amount of this record type in the rare event of a double failure.

This helps determine whether a CF-only log stream is acceptable.

- ▶ Critical record?

This is an indicator of whether this is a critical record type, meaning that you might want to write it to two log streams, use staging data sets, and separate the offload data sets into separate failure domains.

- ▶ LSName

This is for the name of the log stream that you plan to place this record type in.

All the record types in a given log stream must be managed in the same way. That is, they must contain only record types that will be moved to a GDG, or only record types that will be retained in the log stream. All the record types must have the same retention period.

- ▶ GDG or logstream?

This is an indicator showing whether your long-term plan for this record type is to continue moving it to a GDG, or if you will keep the records in the log stream until they expire.

## Stream\_Sizing spreadsheet

The intent of the Stream\_Sizing spreadsheet is to help to size your log streams, based on information about the actual volume of SMF records being created on your system, and your assignment of record types to log streams.

Based on an IFASMFDP report, the spreadsheet attempts to estimate the volume of SMF records that will be created for one day. The spreadsheet then allows you to assign each SMF record type to different log streams, thereby calculating the volume of data that will be sent to each log stream. You can also adjust certain log stream parameters to see the impact.

To get the maximum value from the spreadsheet, run an IFASMFDP job against SMF data sets containing SMF records for all systems in the sysplex for a peak interval. Alternatively, you can run the job against a data set containing SMF data from just one system, and then use the scaling factor (in cell L3 of the Sizing worksheet) to estimate the volume of SMF data that will be created by the entire sysplex.

### Using the Stream\_Sizing spreadsheet

There are a number of steps required to get the information that you need from the Stream\_Sizing spreadsheet.

#### ***Sheet 1: Imported data***

The first step is to import information about the volumes of SMF data that is being generated into the spreadsheet. The CSV file created by the SMFRCSV REXX program (described in “SMFRCSV program: Converting IFASMFDP output to CSV format” on page 141) contains this information.

To import this information into the Stream\_Sizing spreadsheet:

1. Open the Stream\_Sizing spreadsheet.
2. Place the cursor in cell A1.
3. Select **Data** → **Import External Data** → **Edit Text Import**.

The CSV file is delimited by commas and needs to start at cell A1 on the Imported Data worksheet.

### ***Sheet 2: Assign log streams***

The information from the imported CSV file is automatically populated to the Assign Log streams worksheet. You see that column G (Assigned Logstream) contains S01. If you select any cell in column G, you are presented with a selection list that lets you assign each record type to a log stream (S01 to S32). Using the information from the Manage\_SMF spreadsheet (where you assigned each record type to a log stream), assign each record to a log stream name. You must have a default log stream (one that will contain record types that are not assigned to any other log stream), so it is probably a good idea to use S01 for that log stream (because every record type is initially assigned to that log stream in the worksheet).

### ***Sheet 3: Sizing***

The final worksheet (Sizing) sums the volume of records and bytes that will be sent to each log stream, based on your assignments in the Assign Log streams worksheet.

Column F in the Sizing sheet contains a more meaningful name for each log stream. To avoid confusion, take a minute to fill in the log stream names that you will use at this point.

The detailed meanings of the columns in the Sizing sheet are:

**Note:** The first four columns are locked, so that you cannot accidentally overwrite them.

- ▶ **Assigned stream**  
The log streams that were assigned in the Assign Log stream sheet.
- ▶ **AVG Record Length**  
The average record length for this log stream.
- ▶ **Total Number of Records**  
The total number of SMF records expected to be written to this log stream on a daily basis, extrapolated from the IFASMFDP report that was imported to the Imported Data sheet.
- ▶ **Total Number of Bytes**  
The total number of bytes expected for one day for this log stream, again extrapolated from the IFASMFDP report that was imported to the Imported Data sheet.

**Note:** The next set of columns are unlocked, because you need to provide the values for these cells.

- ▶ **Log Stream Name**  
Use this column to specify the actual name that you will use for this log stream. Follow the log stream naming guidelines described in 4.2.2, “Which type of log stream to use” on page 50.



► Log stream type

For each log stream, select the type of log stream that you plan to use:

- DASDONLY
- CF-only
- CF+staging.

► Number of days records will be kept in the log stream

This column specifies how many days you plan to keep the SMF records in this log stream. It should reflect your plans for archiving the records on an hourly or daily basis, or if you plan to keep the records in the log stream until they expire.

► MAXBUFSIZE

This is the block size that SMF uses for writing the data to the log stream. The suggested value is 65532. If this is a CF-only or a CF+staging log stream, this value is one of the parameters that you will input to the CFSIZER to determine the structure size.

► LS\_Size

This specifies the size, in 4 KB blocks, of the log stream offload DASD data sets for this log stream. The valid LS\_SIZE values range from 16 to 1048500.

Attempt to use a value that results in the value in column Q (Expected number of offload data sets per day) being not less than 1.

► HIGHOFFLOAD

This specifies the percentage full (of the CF structure, or the staging data set) at which an offload will be started.

We suggest using a value that is no greater than 60%.

► LOWOFFLOAD

This specifies the threshold at which point the offload will finish.

We suggest using 0 as the low offload threshold.

► Number of days to keep offload data sets on Primary before migration

This specifies how long the offload data sets will reside on primary DASD before they will be migrated by HSM. This value depends on the management class assigned to these data sets and should reflect your planned usage of that log stream.

**Note:** The remaining columns are locked, so that you cannot accidentally overwrite them.

► Total Size of logstream

This is the total amount of data that is expected to reside in this log stream, based on the specified RETPD.

► Primary DASD GB needed for offload data sets

Based on the number of days that you indicated that the data will reside on primary DASD before being migrated, this is the minimum amount of primary DASD needed for this stream (in GBs). This value does *not* include space for offload data sets that are subsequently recalled from HSM for read requests against this log stream.

► Expected number of offload data sets per day

This is the number of offload data sets expected to be generated in one day, based on the offload data set size.

- Write Rate Blks/sec

This is the average write rate for SMF to write blocks to this log stream. The value is averaged over the entire day, so you might have peaks in certain intervals for certain log streams. This value is one of the parameters for CFSIZER, if you plan to use CF structure-based log streams.

- DSEXTENT

Based on the projected number of offload data sets, the spreadsheet calculates the number of DSEXTENTs in the LOGR CDS that will be used for this log stream. This value can be used to ensure that the LOGR CDS is defined with a sufficiently large DSEXTENT value. In addition, 20% should be added to this value, because total DSEXTENT usage should be less than 85%.

## z/OS tools

The `z0Stool1s` zip file contains the following data sets:

- RBITSO.CNTL.XMIT

The JCL to run the sample tools, along with the source for those tools

- SMFDPUX.V2R1M1.CNTL.XMIT

Source and related jobs for the sample IFASMFDP exits

These files should be sent in binary format to FB80 sequential data sets on z/OS, and a TSO RECEIVE run against those data sets.

### RBITSO.CNTL.XMIT data set

The RBITSO.CNTL.XMIT data set should be uploaded in binary format to an FB80 sequential data set on z/OS. Then run a TSO RECEIVE against that data set. The resulting PDS contains a number of members. The members contain a combination of program source (both REXX and Assembler), JCL, and sample reports. The \$INDEX member contains a brief description of the purpose of each member.

### SMFDSACC program

The SMFDSACC REXX program processes SMF record types 14, 15, and 62, and creates three reports:

- A list of executions of the SMF dump program (IFASMFDP). The report shows the input and output data sets for each run of the IFASMFDP program.
- A list of other jobs and programs that access the SYS1.MAN data sets directly. This report shows any program (other than IFASMFDP) that accessed SYS1.MAN data sets. Such programs need to be replaced if you move to logstream mode.
- A list of all jobs that accessed the output data sets that were listed in the two first reports.

The JCL to run the SMFDSACC program is contained in member SMFDSACE and is shown in Example E-1.

*Example: E-1 Sample JCL to run program SMFDSACC*

---

```
//LENNARTA JOB (0,0),'SMFDSACC',CLASS=A,  
//          MSGCLASS=X,NOTIFY=&SYSUID.  
//REXX      EXEC PGM=IKJEFT01,DYNAMNBR=25
```

```
//SYSPROC DD DISP=SHR,DSN=LENNART.RBITSO.CNTL
//SMFDUMP DD DISP=SHR,DSN=SMF.DUMP.ZT00PLEX.G0400V00,
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//          DD DISP=SHR,DSN=SMF.DUMP.ZT00PLEX.G0401V00,
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//          DD DISP=SHR,DSN=SMF.DUMP.ZT00PLEX.G0402V00,
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//          DD DISP=SHR,DSN=SMF.DUMP.ZT00PLEX.G0403V00,
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//WORK14 DD DISP=(,PASS),
//          SPACE=(CYL,(150,25),RLSE),UNIT=SYSALLDA,
//          DCB=(RECFM=VB,LRECL=255,BLKSIZE=0)
//WORK15 DD DISP=(,PASS),
//          SPACE=(CYL,(150,25),RLSE),UNIT=SYSALLDA,
//          DCB=(RECFM=VB,LRECL=255,BLKSIZE=0)
//WORK62 DD DISP=(,PASS),
//          SPACE=(CYL,(150,25),RLSE),UNIT=SYSALLDA,
//          DCB=(RECFM=VB,LRECL=255,BLKSIZE=0)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//REPORTS DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=13304)
//SYSTSIN DD *
//          %SMFDSACC
/*
```

---

The three reports are printed to the REPORTS DD data set. An example of the reports is shown in 2.2, “Documenting your current SMF processes” on page 30, and is contained in member SMFSDACO of the RBITSO.CNTL data set.

**Note:** The REXX program is unable to read VBS data sets as VBS. You need to specify RECFM=VB, and the program will assemble parts of records to full records. This is not a fool-proof method, but is good enough for the purposes of this program.

### SMFRCSV program: Converting IFASMFDP output to CSV format

The SYSPRINT output from IFASMFDP contains statistics about the type, number, and size of the SMF records in the input file that was processed. This information is required for your log stream sizing exercise. To more easily import this information into a spreadsheet, we have provided a sample REXX program called SMFRCSV that creates a data set in CSV format from your SYSPRINT output. The JCL to run the SMFRCSV program is shown in Example E-2 and is contained in member SMFRCSVE.

*Example: E-2 Sample JCL to run program SMFRCSV*

---

```
//LENNARTA JOB (0,0),'SMFRATE',CLASS=A,
//          MSGCLASS=X,NOTIFY=&SYSUID
//REXX EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=50M
//SYSPROC DD DISP=SHR,DSN=LENNART.RBITSO.CNTL
//SMFDUMP DD DISP=SHR,DSN=SMF.DUMP.ZT00PLEX,
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//CSV DD DSN=LENNART.SMFSTAT.CSV,DISP=(,CATLG),
//          SPACE=(TRK,(2,2),RLSE),UNIT=SYSALLDA,
//          DCB=(RECFM=VB,LRECL=500,BLKSIZE=10004)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
```

```
//SYSTSIN DD *
%SMFRATE
/*
```

You can download the CSV data set to your PC in text format (and into a file with a file type of CSV) and open it in a spreadsheet program (Figure E-1).

SUMMARY ACTIVITY REPORT						
START DATE-TIME						END DATE-TIME
RECORD TYPE	RECORDS READ	PERCENT OF TOTAL	AVG. RECORD LENGTH	MIN. RECORD LENGTH	MAX. RECORD LENGTH	RECORDS WRITTEN
2	0					8
3	0					8
14	18895	4.03	379.87	318	576	18895
15	6165	1.32	389.58	318	452	6165
16	188	0.04	817.74	608	1256	188
17	2654	0.57	100	100	100	2654
18	108	0.02	144	144	144	108
21	6	0	88	88	88	6
23	1	0	258	258	258	2
26	2	0	455	455	455	2
30	31093	6.64	1469.02	400	32740	31093
32	3	0	380	380	380	3
33	133	0.03	342	342	342	133
38	467	0.1	208	208	208	467
41	788	0.17	146.74	146	292	788

Figure E-1 IFASMFDP SYSPRINT output in spread sheet

## SMFRATE program: Identifying SMF record volumes

The Stream\_Sizing spreadsheet provides the write rate per log stream, based on the IFASMFDP report that is input to that spreadsheet. However, that write rate is based on the average write rate over the period of the IFASMFDP report. If your workload is known to create SMF records in a “spiky” manner, you might want to know the verify the peak write rate over a shorter interval.

To assist you with this, the SMFRATE REXX program analyzes the SMF records in a data set and provides a report (in CSV format) that contains the number of records and bytes for each record type that are created every hour. It also provides the write rate in the peak second for each record type. The JCL to run the SMFRATE program is shown in Example E-3 and is contained in member SMFRATEE.

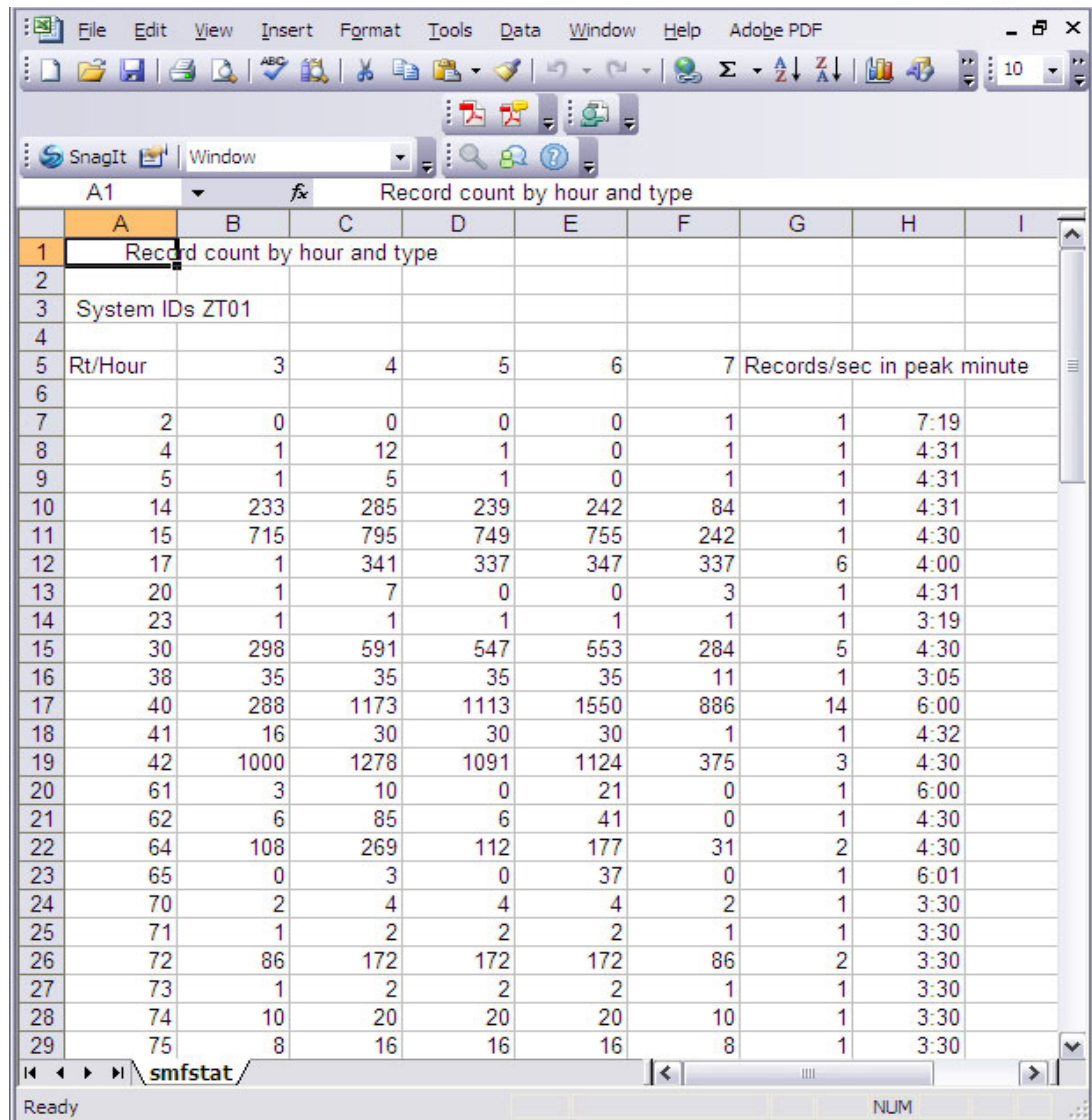
Example: E-3 Sample JCL to run program SMFRATE

```
//LENNARTA JOB (0,0),'SMFRATE',CLASS=A,
//          MSGCLASS=X,NOTIFY=&SYSUID
//REXX      EXEC PGM=IKJEFT01,DYNAMNBR=25,REGION=50M
//SYSPROC   DD DISP=SHR,DSN=LENNART.RBITSO.CNTL
//SMFDUMP   DD DISP=SHR,DSN=SMF.DUMP.ZTOOPLEX(0),
```

```
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//CSV      DD DSN=LENNART.SMFRATE.CSV,DISP=(,CATLG),
//          SPACE=(TRK,(2,2),RLSE),UNIT=SYSALLDA,
//          DCB=(RECFM=VB,LRECL=500,BLKSIZE=10004)
//SYSPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
           %SMFRATE
/*
```

**Note:** Consider compiling the SMFRATE program in order to reduce the run time. In our environment, the compiled program ran three times faster than the interpretive REXX.

You can download the resulting CSV data set to your PC and open it with a spreadsheet program (Figure E-2). A sample report is contained in member SMFRATEO.



	A	B	C	D	E	F	G	H	I
1	Record count by hour and type								
2									
3	System IDs ZT01								
4									
5	Rt/Hour	3	4	5	6	7	Records/sec in peak minute		
6									
7	2	0	0	0	0	1	1	7:19	
8	4	1	12	1	0	1	1	4:31	
9	5	1	5	1	0	1	1	4:31	
10	14	233	285	239	242	84	1	4:31	
11	15	715	795	749	755	242	1	4:30	
12	17	1	341	337	347	337	6	4:00	
13	20	1	7	0	0	3	1	4:31	
14	23	1	1	1	1	1	1	3:19	
15	30	298	591	547	553	284	5	4:30	
16	38	35	35	35	35	11	1	3:05	
17	40	288	1173	1113	1550	886	14	6:00	
18	41	16	30	30	30	1	1	4:32	
19	42	1000	1278	1091	1124	375	3	4:30	
20	61	3	10	0	21	0	1	6:00	
21	62	6	85	6	41	0	1	4:30	
22	64	108	269	112	177	31	2	4:30	
23	65	0	3	0	37	0	1	6:01	
24	70	2	4	4	4	2	1	3:30	
25	71	1	2	2	2	1	1	3:30	
26	72	86	172	172	172	86	2	3:30	
27	73	1	2	2	2	1	1	3:30	
28	74	10	20	20	20	10	1	3:30	
29	75	8	16	16	16	8	1	3:30	

Figure E-2 SMFRATE report

## DELDUPS job

There might be situations in which duplicate SMF records get written to a data set. You can use the sample ICETOOL job shown in Example E-4 to delete duplicate records from the file.

*Example: E-4 ICETOOL job to delete duplicate SMF records*

---

```
//KYNFES JOB (0,0),'DEL DUPS',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//STEP0010 EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//IN DD DSN=KYNFES.SMF.DUPS,DISP=SHR
//OUT DD DSN=KYNFES.SMF.NODUPS,DISP=(,CATLG),UNIT=SYSDA,
//* SPACE=(CYL,(500,100),RLSE)
//TOOLIN DD *
SELECT FROM(IN) TO(OUT) ON(1,999,CH) FIRST USING(CTL1)
/*
//CTL1CNTL DD *
OPTION VLSHRT
/*
```

---

## SMFTYP7

If you want to monitor for SMF discarding records because of a buffer full condition, collect and print the type 7 records. Information about the contents of the record can be found in *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

To get you started, we have provided a simple ICETOOL job that will print the time, date, and number of records that were deleted, as reported in each type 7 record. Note that if no records are deleted, then no type 7 records will be created, and the job will end with return code 16.

The job is in the RBITSO.CNTL data set as member SMFTYP7, and is shown in Example E-5.

*Example: E-5 Job to print SMF Type 7 records*

---

```
//KYNFR JOB (0,0),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*****
/* THIS JOB EXTRACTS SMF TYPE 7 RECORDS FROM A LOG STREAM AND
/* USES ICETOOL TO PRINT THE DATE AND TIME FOR EACH RECORD, AND THE
/* NUMBER OF RECORDS THAT WERE DISCARDED DUE TO THE SMF BUFFERS
/* BEING FULL.
//*****
/*
//S1 EXEC PGM=ICETOOL
//*****
/* UPDATE THE RAWSMF DD STATEMENT TO POINT AT THE DATA SET OR
/* LOG STREAM THAT CONTAINS THE SMF RECORDS.
//*****
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//RAWSMF DD DSN=IFASMF.#@3.TEST,DISP=SHR,
// DCB=(RECFM=VB,BLKSIZE=32760,LRECL=32756),
// SUBSYS=(LOGR,IFASEXIT,'FROM=(2010/349),TO=(2010/365),LOCAL')
//SMF# DD DSN=&&S1,UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(,PASS)
//SMF#REP DD SYSOUT=*
//TOOLIN DD *
```

```

COPY FROM(RAWSMF) TO(SMF#) USING(SMFI)
/*
//SMFICNTL DD *
  INCLUDE COND=(6,1,BI,EQ,07)
/*
//S2      EXEC  PGM=ICETOOL
//*****
///* UPDATE THE SMFICNTL STATEMENTS TO CONTAIN THE NAME
///* OF THE SYSTEM THAT YOU ARE INTERESTED IN.
//*****
//TOOLMSG DD  SYSOUT=*
//DFSMSG  DD  SYSOUT=*
//RAWSMF   DD  DISP=SHR,DSN=*&S1
//SMF#     DD  DSN=*&T1,UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(,PASS)
//SMF#REP  DD  SYSOUT=*
//TOOLIN   DD  *
COPY FROM(RAWSMF) TO(SMF#) USING(SMFI)
DISPLAY FROM(SMF#) LIST(SMF#REP) -
  TITLE('SMF TYPE 7 RECORDS') DATE TIME PAGE -
  HEADER('SMF#') ON(6,1,BI) -
  HEADER('TIME') ON(7,4,TM1,E'99:99:99') - C'HH:MM:SS'
  HEADER('DATE') ON(11,4,DT3,E'9999-999') - C'YYYY-DDD'
  HEADER('SYS') ON(15,4,CH) -
  HEADER('RECS LOST') ON(19,2,BI) -
  BLANK
/*
//SMFICNTL DD *
  OPTION VLSHRT
  INCLUDE COND=(15,4,CH,EQ,C'SYSA')
/*

```

---

## Implementing the sample IEFU29L exit

Because IBM currently does not provide a sample IEFU29L exit, we included one in this book. The sample exit actually consists of an Assembler stub that calls a REXX program, passing the names of the log streams as parameters. The sample REXX program simply returns the log stream names to the console, but you can modify it as you require. Example E-6 shows a sample of the output.

*Example: E-6 Invocation of the sample IEFU29L exit*

---

```

D SMF
IFA714I 18.35.11 SMF STATUS
          LOGSTREAM NAME          BUFFERS          STATUS
          A-IFASMF.STRIPE.TYPDFLT          37485          CONNECTED
          A-IFASMF.##$#PLEX.TYPRMF          12464          CONNECTED
I SMF
IFA705I SWITCH SMF PROCESS HAS SYNCHRONIZED THE BUFFERED LOGSTREAM
          RECORDS.
          * * * * *
          *
          *          SMF LOGSTREAM SWITCH EXIT          *
          *
          * * * * *

```

```
LsName(1): IFASMF.STRIPE.TYPDFLT
LsName(2): IFASMF.#@$#PLEX.TYPRMF
```

---

### ***Installing and activating the IEFU29L exit***

The use of the IEFU29L exit is discussed in 5.1.9, “Install and activate the SMF IEFU29L exit” on page 104.

To install the IEFU29L exit, follow these steps:

1. Assemble and link IEFU29L to a library in your linklist concatenation. The exit needs to be reentrant and reusable. The U29LJOB member in RBITSO.CNTL contains a JCL to assemble and link the Assembler source in member U29LASM.
2. Update your PROGxx member for the dynamic exit facility as follows:  

```
EXIT ADD,EX(SYSSTC.IEFU29L),MODNAME(IEFU29L),DSNAME(your.apf.library)
```
3. You can activate the definition by using either the **SET PROG=xx** command or the **SETPROG EXIT** command.
4. To allow the system to invoke IEFU29L, the exit needs to be defined in SMFPRMxx. Use the EXITS option of the SUBSYS parameter for the STC subsystem. If you have not defined a SUBSYS parameter, you can specify IEFU29L on the EXITS option in the SYS parameter.

Exit IEFU29L needs to be defined in both parmlib members PROGXX and SMFPRMxx if you use the dynamic exit facility. The exit will not be invoked if it is only defined in the SMFPRMxx member.

### ***Making the REXX exec available to SYSREXX***

Copy the U29LREXX exec from RBITSO.CNTL to a PDS in your SYSREXX concatenation. The SYSREXX concatenation is defined in the AXRxx parmlib member. Example E-7 shows a sample.

*Example: E-7 Sample AXRxx member*

---

```
CPF('REXX&SYSCLONE.',SYSPLEX)
AXRUSER(AXRUSER)
REXXLIB ADD DSN(ZS9004.TRAINER.REXXEXEC)
REXXLIB ADD DSN(SYS1.SAXREXEC) VOL(&SYSR1.)
```

---

### ***Verifying the invocation of the SYSREXX U29LREXX***

Issue an **I SMF** command when the system is in logstream mode, and check syslog for messages similar to those shown in Example E-8.

*Example: E-8 Sample output from IEFU29L exit*

---

```
* * * * *
*
*          SMF LOGSTREAM SWITCH EXIT
*
* * * * *
LsName(1): IFASMF.#@$3.PERFT2.LS1
```

---



### ***Customizing the SYSREXX U29LREXX***

When you have verified that U29LREXXexec is being called as a result of the **I SMF** command, customize this REXX to perform the tasks that you want. For example, to start a procedure, add the following line of code:

```
Call Axrcmd "S MYPROC"
```

## **Sample IFASMFDP exits**

While the standard IFASMFDP and IFASMFDPDL reports provide summary information about the SMF records that they encountered, there are times when more detailed information is helpful. The three exits delivered in the SMFDPUX1.V2R1M1.JCL data set and discussed in this section can provide that information.

The SMFDPUX1.V2R1M1.JCL.XMIT file should be uploaded in binary format to an FB80 sequential data set on z/OS. Then run a TSO RECEIVE against that data set. The resulting PDS contain a number of members. The purpose of each member is described in the \$\$README member, and the installation instructions for the exits are contained in the \$INSTALL member.

### **Report formats for input (SMFDPUX1) and output (SMFDPUX2) exits**

The first two exits (SMFDPUX1 and SMFDPUX2) provide a more detailed description of the records contained in each input file (SMFDPUX1) and the records written to each output file (SMFDPUX2).

The type of information provided in the reports that these programs produce include:

- ▶ The number of bytes per record.
- ▶ The minimum and maximum record length of each record.
- ▶ The time and date of the earliest and latest record, at the record type level.

If the record type supports subtypes, all the information above is provided at the subtype level.

### **SMFDPUX3 exit**

In addition to using this information to verify that your test jobs are behaving as you expect, it might be helpful to have this type of information for all SMF processing. If you are interested in this capability, the SMFDPUX3 exit can be used to create user SMF records containing all the information contained in the SMFDPUX1 and SMFDPUX2 reports, so you can actually create SMF records about SMF record processing.

A program (SMFUXPP) to format the user SMF records created by the SMFDPUX3 exit is provided as part of the sample exits.

### **Installing the IFASMFDP sample exits**

The actions required to make the IFASMFDP exits available on your system are described in the \$INSTALL member of the SMFDPUX1.V2R1M1.JCL data set. You also need to enable the exits using the SMFDPEXIT and SMFDLEXIT statements in your SMFPRMxx member.

Once this has been done, all that is required to use the exits is to add three keywords to your IFASMFDP and IFASMFDL jobs. Example E-9 contains sample JCL showing the use of these keywords (USER1, USER2, USER3).

Example: E-9 Sample to invoke the exits using IFASMFDP

```

//*****
//*
//* IFASMFDP - TESTS SMFDPUX USER EXITS
//*
//*          THE FOLLOWING JCL VARIABLES MUST BE SET
//*
//  SET STEPLIB=MARIO.SMFDPUX.V2R1M1.LOAD  SMFDPUX LOADLIB (APF AUTH) *
//  SET SMFIN=MARIO.SMFDUMP.ALLSMF.DATA    IFASMFDP INPUT FILE      *
//*
//*****
//*
//STEP1      EXEC PGM=IFASMFDP
//STEPLIB DD  DISP=SHR,DSN=&STEPLIB
//SMFIN DD    DISP=SHR,DSN=&SMFIN
//SMFOUT DD   DUMMY
//SYSPRINT DD  SYSOUT=*
//SYSIN DD    *
              INDD(SMFIN,OPTIONS(DUMP))
              OUTDD(SMFOUT,TYPE(0:255))
              USER1(SMFDPUX1)
              USER2(SMFDPUX2)
              USER3(SMFDPUX3)
/*

```

Example E-10 contains a sample of the standard report that is created by IFASMFDP.

Example: E-10 Sample IFASMFDP report

SUMMARY ACTIVITY REPORT							
START DATE-TIME 07/29/2010-15:04:18				END DATE-TIME 07/29/2010-17:22:42			
RECORD	RECORDS	PERCENT	AVG. RECORD	MIN. RECORD	MAX. RECORD	RECORDS	
TYPE	READ	OF TOTAL	LENGTH	LENGTH	LENGTH	WRITTEN	
2	0					1	
3	0					1	
14	35	.15 %	354.40	344	608	35	
15	37	.16 %	344.75	344	372	37	
17	1	.00 %	100.00	100	100	1	
23	13	.06 %	258.00	258	258	13	
30	448	1.93 %	1,300.87	400	3,925	448	
32	3	.01 %	232.00	224	248	3	
41	9	.04 %	412.00	412	412	9	
42	323	1.39 %	493.06	176	7,948	323	
60	1,724	7.44 %	546.71	338	547	1,724	
61	5	.02 %	464.40	304	958	5	
62	35	.15 %	188.00	188	188	35	
64	1,713	7.40 %	462.00	462	462	1,713	
65	1	.00 %	304.00	304	304	1	
70	18	.08 %	11,472.00	852	22,092	18	
71	9	.04 %	1,920.00	1,920	1,920	9	
72	576	2.49 %	1,389.43	1,108	13,260	576	
73	9	.04 %	21,728.00	21,728	21,728	9	
74	657	2.84 %	19,750.79	364	32,712	657	
75	27	.12 %	264.00	264	264	27	
77	9	.04 %	337.77	320	480	9	
78	18	.08 %	11,780.00	1,888	21,672	18	
88	129	.56 %	232.79	161	308	129	
89	18	.08 %	627.00	307	919	18	
90	1	.00 %	164.00	164	164	1	
92	107	.46 %	212.00	212	212	107	
201	8,070	34.85 %	14,355.26	64	32,640	8,070	
210	7,342	31.70 %	14,360.44	64	32,640	7,342	
218	1,821	7.86 %	14,155.77	64	32,640	1,821	

TOTAL	23,158	100 %	11,402.51	64	32,712	23,160
NUMBER OF RECORDS IN ERROR			0			

Example E-11 contains a sample of the detailed report the SMF exits produce. The example shown was for the output data set. The SMFDPUX1 exit produces a similar detailed report for each input file.

*Example: E-11 Output from IFASMFDP user exits*

DETAILED ACTIVITY REPORT FOR OUTPUT DDNAME: SMFOUT											
RECORD TYPE	RECORD SUBTYPE	SYSTEM ID	RECORDS WRITTEN	BYTES WRITTEN	RECORD MINLEN	RECORD MAXLEN	START DATE	START TIME	END DATE	END TIME	
2	N/A	#@\$A	1	18	18	18	10.210	17:23:43.50	10.210	17:23:43.50	
3	N/A	#@\$A	1	18	18	18	10.210	17:23:51.17	10.210	17:23:51.17	
14	N/A	#@\$A	35	12404	344	608	10.210	15:04:18.55	10.210	17:21:49.04	
15	N/A	#@\$A	37	12756	344	372	10.210	15:05:29.28	10.210	17:22:09.05	
17	N/A	#@\$A	1	100	100	100	10.210	15:28:10.01	10.210	15:28:10.01	
23	N/A	#@\$A	13	3354	258	258	10.210	15:13:24.32	10.210	17:13:24.32	
30	1	#@\$A	3	1221	400	413	10.210	15:04:18.51	10.210	17:19:23.27	
30	2	#@\$A	331	450861	1119	3925	10.210	15:15:00.00	10.210	17:21:39.20	
30	3	#@\$A	5	6361	1209	1435	10.210	15:06:17.14	10.210	16:35:31.19	
30	4	#@\$A	5	8341	1119	2109	10.210	15:06:17.14	10.210	16:35:31.20	
30	5	#@\$A	5	8393	1119	2122	10.210	15:06:17.14	10.210	16:35:31.20	
30	6	#@\$A	99	107613	1087	1087	10.210	15:15:00.00	10.210	17:15:00.00	
32	1	#@\$A	3	696	224	248	10.210	15:15:11.51	10.210	17:21:39.20	
41	3	#@\$A	9	3708	412	412	10.210	15:13:46.31	10.210	17:13:46.33	
42	1	#@\$A	6	1056	176	176	10.210	15:13:25.58	10.210	17:13:26.39	
42	2	#@\$A	126	32040	212	436	10.210	15:15:00.00	10.210	17:15:00.03	
42	5	#@\$A	9	57708	5980	7948	10.210	15:15:00.00	10.210	17:15:00.00	
42	6	#@\$A	158	61976	308	1604	10.210	15:04:18.42	10.210	17:22:09.05	
42	24	#@\$A	24	6480	270	270	10.210	15:10:52.23	10.210	17:22:09.05	
60	N/A	#@\$A	1724	942538	338	547	10.210	15:04:18.41	10.210	17:22:40.48	
61	N/A	#@\$A	5	2322	304	958	10.210	15:04:18.54	10.210	17:19:41.31	
62	N/A	#@\$A	35	6580	188	188	10.210	15:04:18.55	10.210	17:09:50.35	
64	N/A	#@\$A	1713	791406	462	462	10.210	15:04:18.41	10.210	17:22:40.49	
65	N/A	#@\$A	1	304	304	304	10.210	15:28:10.01	10.210	15:28:10.01	
70	1	#@\$A	9	198828	22092	22092	10.210	15:15:00.01	10.210	17:15:00.01	
70	2	#@\$A	9	7668	852	852	10.210	15:15:00.03	10.210	17:15:00.03	
71	1	#@\$A	9	17280	1920	1920	10.210	15:15:00.02	10.210	17:15:00.02	
72	3	#@\$A	567	680976	1108	1820	10.210	15:15:00.02	10.210	17:15:00.02	
72	4	#@\$A	9	119340	13260	13260	10.210	15:15:00.05	10.210	17:15:00.05	
73	1	#@\$A	9	195552	21728	21728	10.210	15:15:00.01	10.210	17:15:00.01	
74	1	#@\$A	171	5391216	11648	32632	10.210	15:15:00.01	10.210	17:15:00.02	
74	2	#@\$A	9	76824	8536	8536	10.210	15:15:00.05	10.210	17:15:00.05	
74	3	#@\$A	9	4104	456	456	10.210	15:15:00.05	10.210	17:15:00.05	
74	4	#@\$A	18	131040	6416	8144	10.210	15:15:00.17	10.210	17:15:00.15	
74	5	#@\$A	414	7202664	1476	32712	10.210	15:15:00.23	10.210	17:15:00.27	
74	6	#@\$A	9	3276	364	364	10.210	15:15:00.17	10.210	17:15:00.15	
74	8	#@\$A	27	167148	4052	10044	10.210	15:15:00.27	10.210	17:15:00.27	
75	1	#@\$A	27	7128	264	264	10.210	15:15:00.02	10.210	17:15:00.02	
77	1	#@\$A	9	3040	320	480	10.210	15:15:00.02	10.210	17:15:00.02	
78	2	#@\$A	9	16992	1888	1888	10.210	15:15:00.02	10.210	17:15:00.02	
78	3	#@\$A	9	195048	21672	21672	10.210	15:15:00.02	10.210	17:15:00.02	
88	1	#@\$A	63	19404	308	308	10.210	15:15:00.00	10.210	17:15:00.00	
88	11	#@\$A	66	10626	161	161	10.210	15:13:48.51	10.210	17:15:00.00	
89	1	#@\$A	9	3015	307	391	10.210	15:15:00.00	10.210	17:15:00.00	
89	2	#@\$A	9	8271	919	919	10.210	15:15:00.00	10.210	17:15:00.00	
90	N/A	#@\$A	1	164	164	164	10.210	15:04:18.42	10.210	15:04:18.42	
92	11	#@\$A	107	22684	212	212	10.210	15:04:54.50	10.210	17:22:42.58	
201	1	#@\$A	8070	115846976	64	32640	10.210	15:04:18.20	10.210	16:35:30.10	
210	1	#@\$A	7342	105434368	64	32640	10.210	15:04:18.31	10.210	16:06:51.33	
218	1	#@\$A	1821	25777664	64	32640	10.210	15:04:18.24	10.210	15:31:18.29	

The PPSGUIDE member of the SMFDPUX1.V2R1M1.JCL data set contains a PowerPoint slideshow with more information about the exits and their use and control. Download that member to your PC to view it.

It might be easier to analyze the information in the report if it was in a spreadsheet. You are able to generate a spreadsheet report from the SMF records created by SMFDPUX3. Example E-12 contains a sample job that you can run to perform this function. This sample consists of two steps. The first step runs IFASMFDP to extract the user SMF record type 255 records that were created by the SMFDPUX3 exit on previous IFASMFDP runs. The second step reads the type 255 records and produces a CSV file that can be downloaded to your workstation and imported into a spreadsheet.

*Example: E-12 How to create the input for the spreadsheet*

---

```
//Jobname JOB ACCT
//EXTR EXEC PGM=IFASMFDP
//STEPLIB DD DISP=SHR,DSN=HLQ.SMFDPUX.V2R1M1.LOAD
//SYSPRINT DD SYSOUT=*
//SMFIN DD DISP=SHR,DSN=SYS1.#@$2.MANC
//SMFOUT DD DSN=&&SMFOUT,DISP=(,PASS),SPACE=(CYL,50),UNIT=SYSDA
//SYSIN DD *
    END(2400)
    START(0000)
    DATE(1900000,2099366)
    USER3(SMFDPUX3)
    USER2(SMFDPUX2)
    USER1(SMFDPUX1)
    OUTDD(SMFOUT,TYPE(255))
    INDD(SMFIN,OPTIONS(DUMP))
/*
//SMFUXPP EXEC PGM=SMFUXPP,PARM='/RECTYPE 255'
//STEPLIB DD DISP=SHR,DSN=HLQ.SMFDPUX.V2R1M1.LOAD
//SMFIN DD DSN=&&SMFOUT,DISP=(OLD,DELETE)
//SYSMSG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//IFAFDATA DD SYSOUT=*
//IFARDATA DD SYSOUT=*
/*
```

---

## How to use the web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the web material zip files into this folder. The spreadsheets can be used as they are. The XMIT files must be uploaded in binary format to FB80 sequential data sets on z/OS, and a TSO RECEIVE run against those data sets.

# 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 publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898
- ▶ *System z Parallel Sysplex Best Practices*, SG24-7817
- ▶ *z/OS Version 1 Release 12 Implementation*, SG24-7853

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

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

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS MVS Setting up a Sysplex*, SA22-7625
- ▶ *z/OS MVS System Commands*, SA22-7627
- ▶ *z/OS MVS System Management Facilities (SMF)*, SA22-7630

## Online resources

The following website is also relevant as a further information source:

- ▶ IBM System z Hot Topics newsletter  
[http://www-03.ibm.com/systems/z/os/zos/bkserv/hot\\_topics.html](http://www-03.ibm.com/systems/z/os/zos/bkserv/hot_topics.html)

## Help from IBM

IBM Support and downloads

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

IBM Global Services

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



# Index

## A

abends  
    possible reasons 96  
accessing the SYS1.MAN data sets directly 23  
active-type log streams 9  
advantages of logstream mode 17  
applications that process SMF records 5  
Approach 1 41  
Approach 2 42, 50  
Approach 3 42, 50  
Approach 4 44, 50  
ARCHIVE  
    comparison to ALL function in IFASMFDP 13  
    considerations for how long data should be kept in the log stream 73  
    function in IFASMFDL 13  
    processing at log block level 39  
attributes of different log stream types 51  
attributes of SMF records best suited to GDGs 43  
AUTODELETE log stream attribute  
    relationship to RETPD attribute 62  
AUTODELETE, use of for CF log streams 62

## B

BUFSIZMAX parameter to control SMF buffer size 3  
BUFSIZMAX, use of to control storage for buffers 68  
BUFUSEWARN enhancements in z/OS 1.12 18  
BUFUSEWARN values 66

## C

catalog messages 79  
catalog recovery 63  
catalog recovery SMF records 100  
CF Sizer tool 59, 61  
CF+staging log stream 50–51  
CF-only log stream 50–51  
CFRM policy 61  
CFSizer 46  
changing log stream attributes 133  
checking offload data set CI Sizes 79  
CI Sizes for Logger data sets 72  
comparative performance results 17  
comparison of ARCHIVE to DUMP function 15  
considerations for migrating offload data sets 14  
considerations for test or development LPARS 50  
considerations regarding which type of log stream to use 51  
Critical records column in Manage\_SMF spreadsheet 31

## D

D SMF command 124  
DASDONLY log stream 50

DCB characteristics 96  
deciding which type of log stream to use 51  
defining CF structures to Logger 61  
defining IFASMFDL/IFASMFDP dump exits in SMF-PRMxx 66  
defining log streams to System Logger 61  
deleting all log blocks from an SMF log stream 131  
deleting duplicate SMF records 132  
determining log stream DASD space requirements 60  
DFSMSHsm and log streams 7  
difference between ARCHIVE and DUMP 111  
differences in processing between ARCHIVE and DUMP 38  
directly accessing SMF sequential files 23  
discarding SMF records 3  
displaying range of dates in a log stream 83  
dumping the SMF address space 129

## E

extended format data sets 5  
    SMF support 41  
extracting SMF records from a dump 129  
extracting SMF records from a dump using IPCS 102

## F

factors that determine log stream size 57  
flood automation support 4  
FULL Enhanced HOLDDATA 80  
funnel-type log streams 9

## G

GDPS considerations 44  
GDPS/PPRC Control systems 23  
getting a list of SMF log streams 83  
granting access to SMF log streams 101–102

## H

handling critical SMF records 46  
HLQ used for SMF log stream offload data sets 63  
how many log streams to have 50  
how programs write records to SMF 2  
HSM recalls for SMF log streams 73

## I

I SMF command 40, 125  
identifying data sets that contain SMF data 32  
identifying SMF-related APARs 79  
identifying who is using SMF records 34  
IEASYMxx member, using to share SMFPRMxx between multiple systems 105  
IEASYMxx member, using with RECORDING value 65  
IEFU29 exit 3, 40, 67

- IEFU29L exit 40, 67, 145
- IFA832I message 131
- IFASEXIT
  - and SMARTENDPOINT 15
  - considerations related to volume of data returned to user program 24
- IFASEXIT program 86
- IFASEXIT restrictions 43
- IFASMF.DUMP00 log stream 102, 129
- IFASMFDL 38
  - ARCHIVE
    - control statement restrictions 39
  - ARCHIVE process 40
  - ARCHIVE processing 131
  - comparison to IFASMFDP 12, 38
  - DUMP compared to ARCHIVE 15
  - options 13
  - RACF considerations 101
  - RELATIVEDATE function 13
  - SMARTENDPOINT function 14
  - using to ease migration to logstream mode for user programs 23
- IFASMFDL program 28
- IFASMFDL return codes 112
- IFASMFDP
  - and duplicate SMF records 12
  - functions 12
  - listing executions of this program 32
- implementation plan objectives 52
- importance of sorting SMF records 114
- important APARs 13–14, 131
- important messages 75
- improving resiliency for SMF log stream structures 52
- introduction to SMF 2
- IPCS batch job 130
- IPCS SMFDATA command 130
- IXG283I message 79
- IXGOFLDS process to initiate log stream offload 78
- IXGSEXIT program 86
- IXGWRITE calls 8

## L

- log block
  - deletion processing 15
  - description 6
  - log block ID 6
  - timestamp values 6
  - timestamps 14
- log stream attribute considerations 62
- log stream naming convention 62
- log stream sizing 57
- log stream types 50
- LOGR couple data set 102
- logstream mode
  - restrictions 44
- logstream mode benefits 17
- Lost records acceptable column in Manage\_SMF spreadsheet 31
- LPAR storage requirements for logstream mode 100
- LSNAME parameter in the SMFPRMxx member 10

## M

- Manage\_SMF spreadsheet 30–31, 35, 38, 50, 57–58, 136
  - obtaining the information for the spreadsheet 34
- mapping SMF record types to sequential data sets 34
- MAXBUFSIZE value for DASDONLY log streams 64
- MAXDORM 16, 132
- MAXDORM time 3
- migration approaches 42
- migration considerations 22
- migration options 16, 28, 40
- migration sequence 44
- monitoring performance of logstream mode 90
- monitoring storage group utilization 109

## N

- NOBUFFS
  - enhancements in z/OS 1.12 18
  - parameter 3
- NOBUFFS keyword in SMFPRMxx 65

## O

- obsolete SMF record types 31
- offload data sets 9
- offload processing 9
- offloading a log stream 78
- OFFLOADRECALL, use with SMF log streams 64
- operator education 74
- optimizing offload performance 110
- OPTIONS(ALL) support 112

## P

- physical deletion of records from a log stream 7
- process to dynamically change log stream attributes 71
- processor storage considerations 24
- programs that access SYS1.MAN data sets directly 33
- programs that process SMF records
  - identifying 28
- PROMPT(xxx) keyword in SMFPRMxx member 64
- protecting access to SMF log streams 101
- protecting critical SMF records 65
- protecting SMF data in log streams 101

## R

- RACF controls 101
- real storage requirements for SMF buffers 68
- reasons for having multiple log streams 45
- Redbooks Web site 151
  - Contact us x
- relationship between log blocks and SMF records 14
- RELATIVEDATE 131
  - use of (BYDAY,0,1) 13
- RELATIVEDATE function 39
- RELATIVEDATE function in IFASMFDL 13, 95
- removing duplicate SMF records 39
- requirement in IFASEXIT program 96
- restricting SMF record collection 4



RETPD log stream attribute  
    relationship to AUTODELETE attribute 62  
RETPD use for log streams 7

## S

sample IFASMF DL ARCHIVE jobs 91  
sample migration strategy 52  
selecting the best log stream type 51  
sharing SMF log stream structures 52  
sizing DASD requirements for SMF log streams 60  
sizing log streams 57  
sizing SMF CF structures 58  
sizing staging data sets 60  
sizing the SMF log streams 57  
SMARTENDPOINT 43, 73  
    relation to ARCHIVE and DELETE processing 15  
    SUBSYS=LOGR with IFASEXIT 15  
    using to limit the volume of SMF data returned to user  
        programs 23  
SMARTENDPOINT function in IFASMF DL 14  
SMF  
    block sizes 12  
    buffers in logstream mode 11  
    buffers when in dataset mode 3  
    critical records 5  
    dataset mode disadvantages 4  
    how it works in dataset mode 2  
    how it works in logstream mode 10  
    how log blocks change retrieval of SMF records 13  
    record handling 5  
    record volumes 4  
    running out of buffers 3  
    scalability 10  
    shared log streams 11  
    spanned blocks 12  
    support for log streams 10  
    support of CF and DASDONLY log streams 12  
    tasks 11  
    timestamp fields 3  
    type 7 records 3  
    use of spanned blocks 3  
SMF log stream naming convention 52  
SMF log stream to structure relationship 58  
SMF real storage considerations 68  
SMF record resiliency considerations 51  
SMFDSACC program 32  
SMFDSACC program 30  
    key to the reports 33  
    role in migration process 30  
SMFDSACC REXX exec 90  
SMFDSACC REXX program 140  
SMFEW TM macro 2  
SMFPRMxx member 64  
SMFRATE REXX exec 59  
SMFRATE REXX program 142  
SMFRCSV REXX exec 57, 141  
SMFRCSV REXX program 137, 141  
SMFW TM macro 2  
SMP/E REPORT MISSINGFIX job 80  
SMS data class 62, 100

SMS data class considerations 72  
SMS management class 73, 100  
SMS messages 79  
SMS storage class 100  
SMS storage group 60  
sorting SMF records 3  
spanned records 3, 12  
statements of direction 41  
storage for SMF buffers 67  
storage group considerations 74  
storage group storage requirements 100  
Stream Sizing spreadsheet 59  
Stream\_Sizing spreadsheet 52, 58, 60, 65, 73, 137  
SUBSYS=LOGR interface 28  
    use with user programs 24  
supported block sizes 12  
switching between logstream and dataset mode 77  
SYS1.MAN data set switching 3, 40  
sysplex-wide log streams 11, 18  
System Logger  
    benefits 6  
    considerations 24  
    functions 6  
    introduction 6  
    log block 13  
    logical vs. physical view of log stream 6  
    offload processing 9  
System Logger abend 113  
System Logger messages 76

## U

understanding SMF record usage 28  
user catalog preparation 100  
using IEBGENER to print SMF records 85

## V

VSAM SHROPTIONS for log stream data sets 72

## Z

Z EOD command 77, 125











**Redbooks®**

# SMF Logstream Mode

## Optimizing the New Paradigm

**Learn the differences  
between logstream  
and dataset modes**

**Understand the  
impact of migrating to  
logstream mode**

**Implement logstream  
mode**

This IBM® Redbooks® publication positions the use of System Logger log streams as a repository for System Management Facilities (SMF) data against the previous use of Virtual Storage Access Method (VSAM) data sets for SMF data. This book expands on existing material by covering not just the implementation steps, but also by looking at how you use SMF data today, and using that information to help you identify the most appropriate repository for your SMF data.

If it transpires that log streams are appropriate for some or all of your SMF data, this book provides all the guidance you are likely to require for a successful migration to this new paradigm.

The target audience for this document is system programmers and anyone that uses SMF data.

**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)**

SG24-7919-00

ISBN 0738435384