

Cheryl Watson's *z/OS 101 Primer*

Updated 2009

The z/OS 101 Primer

At the 2008 SHARE conference held in Orlando, I gave a lunchtime presentation called *The Tips Your Mentor Forgot to Mention*. It was for the zNextGen project, which was created to help those people who are new to mainframes or new to performance. The response to that project was incredible. One teacher even brought several students from his class in Georgia to the conference JUST to attend the zNextGen sessions.

Because of the project's success, and because several of our readers responded so favorably to my advice for new techies, I decided to include a new section in every Tuning Letter called "z/OS 101". Each Tuning Letter issue will contain this section and address a topic that should be useful to those new to mainframes, especially in the performance, capacity planning, data center reporting, and charge back areas. If you'd like some special topic addressed, please let me know.

This particular document is a compilation of the z/OS 101 articles written so far and is offered to the public via our website. Please use it and copy it as you see fit, provided that you use the entire document if you distribute it, always credit us as the source, and make no attempt to resell it. The most recent compilation can always be found at www.watsonwalker.com/zos101.pdf.

So, hats off to our next generation of z/OS techies, and welcome to the exciting world of performance!



Advice for z/OS Techies.....	2
More Advice for New Techies	4
Linda Mooney's Tips	4
And More	7
SMF Introduction	8
MVS Commands	9
SMFPRMxx	10
References	11
SMF Exits	12
Life of an Address Space.....	14
References.....	21
Volume-Related SMF Records .	22
File-Related SMF Records	24
Total I/O Activity.....	27
References.....	28
System Logger.....	29

Advice for New z/OS Techies

As I mentioned, I was on a panel at SHARE called *zNextGen Lunchtime Panel Discussion: The Tips Your Mentor Forgot to Mention*. In preparation for that panel, I was asked to list some of my best advice to people who are new to z/OS. So I wrote up the following list that I hope will be of use to some of our readers. We'll probably have some additions after the panel, and would love to include any of your favorites in our next newsletter. So please send any additional ideas to me at <mailto:technical@watsonwalker.com>.

Advice for New z/OS Techies

1. Measure everything you do! Whether you make a change to a program, a parameter, a file specification, or simply move a file to a different device, you could make things better or you could make things worse. If things got better, your manager should know so that he/she will trust you to make other changes. If things got worse, you can back it out so that your manager doesn't even need to know! If you are trying to improve TSO response time, make sure that you don't penalize the CICS users, or vice versa. This is the most important piece of advice that I can give, and I followed it for over 40 years.
2. Don't ever think that an area of computing doesn't affect you - it does. If you are a systems programmer, what the DB2 application programmers do will impact your system. If you're a DB2 application programmer, how the system programmer sets up parameters will impact your application. If you are only looking at z/OS and not listening to what's happening with Linux on z/VM, you may get behind the curve. I've been a generalist all my life and it's served me well because I can apply what I learn in one area (e.g. DB2 likes lots of storage) to other areas (e.g. CICS also likes lots of storage). It helps me understand the types of things to look for.
3. If you don't like to read manuals, or don't have the time, then often your best option is to GOOGLE it (<http://www.google.com>). If I forget what the ISPF keys are, I can find them quicker with Google than by searching through several manuals (is it in the Users Guide or the Users Reference?). If you only want to find things on the IBM web site, then you can enter 'site:ibm.com' at the end of your search (such as 'smf exit iefujv site:ibm.com') and you'll find APARs, publications, and other goodies to help you out.
4. For helpful links, we have a great handout of links for new users. You can find it on our website at <http://www.watsonwalker.com/PR070213.pdf>. I'm not going to duplicate the contents here - it's several pages long. If you click on a link in the presentation, you can go directly there.
5. One of my favorite sources of information is the Washington Systems Center (with their flashes, presentations, hints & tips, etc.). They have some outstanding pieces of information. Just place a keyword or two in their search at <http://www.ibm.com/support/techdocs>.

6. I think that the best way to understand something is from the IBM Redbooks. They are extremely easy to read and either explain everything from step one, or will at least provide a bibliography to get you started. See <http://www.redbooks.ibm.com>.
7. One redbook I love dates from 1999 - SG24-4757 (P/390 OS/390: New User's Cookbook). It was developed for software developers who might not be MVS sysprogs, and who might need help with some of the basics: How to IPL, How to Reset a Password, How to Add RACF Users, How to Clear SYS1.LOGREC, How to Set Up NJE, etc., etc. What I like most about it is that it's all in one place, and I always keep it by my side. You can find it on the Redbooks site above.
8. Find a good users' forum and become familiar enough that you can ask a question if you need to. Check out our presentation (in #4 above) for several suggestions.
9. Share what you learn. Whether returning from a SHARE conference, or reading something new in a manual, share the information with your colleagues. They will appreciate it, and maybe they'll return the favor. Spreading knowledge instead of hiding it helps everyone. And of course don't forget to share at SHARE. ■

More Advice for New z/OS Techies

If you missed my article in the last Tuning Letter, please start there with my original advice. Here, I'll include some new thoughts that were brought up during the panel or sent in by our readers. Thanks to fellow panelists **Chris Craddock**, **Dave Danner**, and **Bob Rogers**; and thanks to contributors from the audience for the following tips:

- Create a baseline of your system. "If you don't know when things are running well, then you won't know when it's running wrong." Although the panel didn't have time to go into specifics about what a baseline would include, I think that it should certainly include both key system indicators (average and peak CPU utilization by LPAR and system, DASD response time by CU, amount of paging (if any), network traffic), as well as application indicators (average response times, average and peak transaction rates, CPU usage by application, average and peak users).
- Question everything, and read (or try to read) manuals before asking the expert in your installation.
- Become a mentor yourself. This is especially applicable if you have UNIX knowledge that the more senior mainframers don't have.
- Do what you enjoy; let the wind blow your boat; don't do things for career advancement only.
- Trust your instincts and work hard.
- Remember that the mainframe has more users than a workstation or PC. You can't simply reboot. Anything you do will affect more people, especially an outage.
- Use Google for many of your questions. You'll be amazed.
- Put a date on every piece of documentation.
- Explain and document why you're doing something. This applies to program changes, parmlib changes, JCL changes, and more.

Linda Mooney's Tips

Linda Mooney is one of the zNextGen's project managers and self-described 'mentor' and 'mentee'. (Her real job is as a Senior IT Analyst at County of Sacramento.) She was scheduled to appear on the panel, but wasn't able to make it. She's graciously allowed me to include her list of tips here (I think they're wonderful!):

- 1.** Most people who have a lot of expertise tend to forget to mention one or more of the steps that are automatic (for them) when they are describing what or how to do something. Take notes; list everything you need to do. Ask your mentor to check it. Later on, when you are working on your own, these lists will be your cheat sheets.
- 2.** If you can't handle change and you aren't thrilled to see what's next, find another line of work! This business has changed tremendously over the years. Stay for the zNextGen Project opening and listen closely to the presentation. If you just can't stay, be sure to get a copy of the presentation and a copy of the grid of the zNextGen recommended sessions. Associate yourself with the zNextGen project, check out your educational opportunities through the IBM education Initiative, and always be on the lookout for opportunity. Remember, "Opportunity is not a lengthy visitor." (Into the Woods, Steven Sondheim, for rent on DVD)
- 3.** Mentors are almost always overloaded with work. They often don't really have the time – and some don't really want to mentor someone. It is up to you to make it a positive experience for them too! My first systems programmer mentor did me a favor when he told me straight out, I really don't have the time to deal with this. I don't have time for you. I was a good operator at the time, and I could roll COBOL, but I had no systems programming experience and he saw the enormity of what I didn't know more clearly than I did. I asked what it would take to get some time from him. First he taught me how to do his most routine tasks, then his monitoring and product testing. After I took that on and he didn't have to do it anymore, he had time to teach me lots of stuff I wanted to know – and he was happy to see me. As long as I didn't run out of questions, he didn't have to do his routine stuff. By the way, I learned a LOT by doing all of those routine tasks. Take the initiative. Who does the things you want to do in your career? What can you do to learn from them? Come in early to take some of their load? You pay for school don't you? What is one-on-one mentoring worth to you? You can do or be whatever you decide to. Dream, decide, act!
- 4.** Never talk bad about your mentor. If you are unhappy, it isn't working out and you really want to work with someone else, say only that you are seeking additional opportunities and perspectives. If you trash one mentor, do you think another will want to take you on?
- 5.** Ask your mentor why they do things the way they do them, why they use the utilities and use the methods they do. Is it because of product requirements, dependencies with other products or systems, shop standards, or personal preference? Be sure you fully understand. Your mentor may retire or leave the company. You may have to do it yourself next time.
- 6.** Listen to your mentor's IT stories. You will learn a lot. Tell your mentor how much you appreciate their help. Don't you like to be told when you have done well?
- 7.** When you install a product, there will be libraries that you will need to customize. Save yourself some time and a red face. As soon as that library is created, make a copy of it adding a low level qualifier of .VENDOR or

VANILLA, whatever. Don't make any changes to that library and be sure to keep it for the life of the product. If you need to get a fresh copy of a JCL or configuration member, it will be handy.

- 8.** ALWAYS negative test your products. Run all of the fat finger checks a customer or newbie might stumble into. Get your security folks to set you up with an id with limited access so that you can make sure that the product's security calls happen the way they should. ALWAYS run shop-specific work to test your products in addition to the vendor provided IVPs (installation verification programs). Do you want to find your mistakes and any problems with the product itself or do you want your customers to find them? Ask your mentor how to do this for your products.
- 9.** ALWAYS make and keep your own backups. If you get in the absolute habit of building yourself a 'safety net' you won't have to worry about SPLAT!! Think of it as 'face plant in public' avoidance. A mistake or something else unforeseen can result in a moment of inconvenience nobody else has to know about or it can result in system downtime or corrupted production data that ends up on the front page of the local newspaper. Be sure you have a backup, a good backout, and a good recovery plan.
- 10.** ALWAYS check your results for every job you run, no matter how simple, no matter how many times you have run it before. Success with any IT task is a combination of your effort and knowledge, what the system does with what you give it, and your follow up.
- 11.** Condition code 4 is not always okay!! Check out the what, how, and why before you accept CC0004.
- 12.** Do you work with products that install with SMP/e? NEVER, NEVER, NEVER ACCEPT a USERMOD. I said NEVER! If you do, you'll wish you hadn't. Just ask your mentor!

ALWAYS, ALWAYS, use the SMPLOG and SMPLOGA datasets. You can use a separate set for each zone (GLOBAL, TARGET, DLIB) or just one set for each product. For z/OS, I use one for each zone, just like IBM supplies it. For ISV (independent software vendor) products, I use one SMPLOG, SMPLOGA set for each product install - (hlq.product.SMPLOG). That way I have a chronological log of the SMP/e I ran to install and support that product. I save the listings too, not just the SMP/e listings. While you are in TSO/SDSF, type XDC next to the job in the output list, fill in the panel and the listing will be copied to a dataset. Name the listing datasets so that the names tie in with the product dataset names. Save them for the life of the product. If you have to put maintenance on that product next year, you will be glad that you won't have to wonder about how things were done. I do this even though my shop has an online viewing product. I control the retention period on my datasets, so I can keep them as long as I want.

- 13.** Learn to use your tools really well! Be sure to learn how to use the IBM utilities, especially IEBCOPY, IEBGENER, IEHLIST, IEHPROGM, IDCAMS, IEFBR14, and IEBPTPCH. They are very useful and they will always be available where ever you work with z/OS. Explore and learn all about TSO.

- 14.** Be curious. Seek knowledge, Share knowledge, Share at Share! Don't be a Know-It-All! If you are not smart enough to realize that there is so much to do, learn and see in this business that nobody (that includes you) will ever master it all – well, go away kid, ya bother me! However, if you are like most of us, a product of the event driven learning that comes with rising to challenge after challenge, and you realize that your personal knowledge base still has too much free space, you will do well. Welcome aboard!

And More

Some readers sent in their favorite advice as well:

- **Mike Duffy** from Lloyds Bank provided these tips:
 - The new guys should cultivate the good techies at their site whether they are operators, DBAs, sysprogs, or operations support, as they usually have a wealth of experience.
 - Attend user groups and network.
 - Search the Web for mainframe forums (e.g. MXG, CMG etc), and participate.
 - If you get the chance on your system – experiment.

- **Howard Merrill** from Verizon suggested this:

One piece of advice I give to anyone that touches a computer (mainframe, PC....)

 - 1) Make sure you do a backup!
 - 2) Make sure you know how to USE the backup!!!

I have come across many situations in the last, um, 30 years, where someone did do a backup but had no idea how to use it. Take it from a Storage Manager, good advice! ■

SMF Introduction

System Management Facilities, SMF, is the component of z/OS that provides a common receptacle for recording information. SMF provides a set of macros that can be used by applications in order to pass records to SMF for recording to a file. The beginning of all records must conform to a standard format, and each record is identified by a one-character record number (hex value '00' to 'FF') between 0 and 255. IBM has exclusive use of record types 1 through 126, and subsystems and applications may use 127 to 266. Applications are not forced to use SMF for recording, and some don't. Several of the record types also have subtypes, such as the type 30 record which has subtypes 0 through 6 to identify unique types of records.

The basics and use of SMF are documented in the z/OS System Management Facilities (SMF) manual, as are most of the record layouts.

The major elements of SMF are:

- **Macros** – Several macros are used by applications to pass information to SMF and to interrogate parameters. The most common of these is the SMFWTM macro, which is used to pass records to SMF. These are documented in the SMF manual.
- **Parameters** - Parameters are used to determine how SMF is run, and which record types are recorded. The parameters reside in member SMFPRMxx of the system parmlib. They are documented in the MVS Initialization & Tuning Reference, although some additional material is also found in the SMF manual. Our Focus article in our Tuning Letter 2003 No. 3 provides a description and recommendations for all of these parameters.
- **Exits** – User-written exits are available to interrogate, change, or delete records as they are passed to SMF. These are normally defined in the PROGxx parmlib member, and are documented in the SMF Manual (see references). I'll be covering these in Part 3 of the SMF Update, which will be in our next newsletter.
- **MVS Commands** – The 'S SMF' command is used to start SMF recording. And the SETSMF and SET SMF commands can sometimes be used to modify the SMF parameters. See our later description of MVS Commands on page 9.

Major Record Types

Everyone should be familiar with certain SMF records, so if you're new to SMF, here's a shortcut. If you know at least these, you'll sound like you're experienced!

- **Types 70-79** – Created by IBM's RMF (Resource Measurement Facility) or BMC Software's CMF Monitor, and used to provide performance statistics for z/OS. The type 70 contains CPU and LPAR usage, the type 74 provides DASD activity

and response times, and the type 72 provides detail by service class periods. These three records and the type 30 records are the most frequently used records.

- Type 30 – Created by MVS, these are written at key points during the processing of any batch job, TSO user, or started task. A subtype 1 is written at the beginning of the job, subtype 2 is written at the end of an interval (if INTERVAL is turned on), subtype 3 is written as the last record of an interval, subtype 4 is written at the end of a step, and subtype 5 is written at the end of a job. In the early 1990s, this record replaced the older types 4, 5, 20, 34, 35, and 40 records because the fields became too small.
- Type 14-15 – These records are written when non-VSAM files are closed and contain information about the usage of each dataset. You can do a lot of performance tuning by analyzing buffer sizes and activity of datasets with these records.
- Type 60-69 – These contain information about VSAM datasets and catalogs, and can be used to tune VSAM files.

MVS Commands

There are three commands that are used for SMF. Two can modify the SMF settings, and one is used to display the information.

SETSMF

The SETSMF command can be used to override one or more SMF parameters while SMF is active. It cannot be used if NOPROMPT is specified in the SMFPRMxx member that was used to start SMF. It can override any parameter but ACTIVE, PROMPT, SID and EXITS. It can be abbreviated as SS. If overriding MAXDORM or STATUS, the new value won't take effect until the previous interval expires. The format is:

setsmf parameter(value) e.g. ss jwt(0030)

SET SMF

The SET SMF command can be used to replace the SMFPRMxx member that was used when SMF was started, and also can be used to restart SMF. This command may be used even if NOPROMPT is specified, and most operators use the abbreviation of 't'. The format is:

set smf=xx e.g. t smf=s1

All of the parameters can be changed except SID by simply changing them in the SMFPRMxx member itself. Because SYS1.PARMLIB is usually protected with tight

Figure 1 - Sample Documentation for SMFPRMxx

```
/******  
/* SMFPRMxx Change Log: */  
/* 8/1/08 û Started with Cheryl Watson's suggestions from TL08-3 */  
/* */  
/* Vendor Record Types: */  
/* 211 û Vendor Been There, added on 8/2/08 */  
/* 232 û Vendor Done That, added on 8/3/08 */  
/******
```

References

IBM APAR [II07124](#) (INFO, 8Apr2008) - *DB2 DBM1 High CPU Utilization in SMFEXIT IEFTB728 or IEFTB726 CICS Transactions in I/O Wait*. Provides great explanation of the effect of DDCONS, DETAIL, and INTVAL on DB2.

IBM APAR [OA24359](#) (z/OS 1.7+, 7May2008) – *SMF is Using a Lot of Auxiliary Storage Slots. MSGIRA200E and MSGIRA204E Could be Issued*.

IBM, *z/OS V1R9.0 MVS Initialization & Tuning Reference*, [SA22-7592-15](#), for a direct link to z/OS 1.9 SMFPRMxx parameters:

<http://publib.boulder.ibm.com/infocenter/zos/v1r9/topic/com.ibm.zos.r9.ieae200/smfprm.htm#smfprm>

IBM, *z/OS V1R9.0 MVS System Management Facilities (SMF)*, [SA22-7630-15](#)

Merrill, Barry, [MXG Newsletter Forty Seven](#), BUFSIZMAX

Merrill, Barry, [MXG Newsletters 16-18, 20, 23, 31-33](#), DDCONS, DETAIL, INTERVAL

Watson, Cheryl, *PARMLIB Analysis*, Cheryl Watson's Tuning Letter 1996 No. 3, pages 34-43

Watson, Cheryl, *SMF Update – Part 1*, Cheryl Watson's Tuning Letter 2008 No. 2, page 6-9 ■

Also, you can find a summary of SMF record types in our SMF Reference Summary at <http://www.watsonwalker.com/SMFreference.pdf>. I plan to update this in 2009. ■

SMF Exits

There are several points during the life of an address space when a user may access the address space information and SMF records using SMF exits. In this section, I'd like to describe the exits, and how they are used.

In Tuning Letter 2008 No. 3, I described the SMF parameters. Two of the parameters were SYS and SUBSYS that specify the type of records that you want to collect. One of the subparameters is EXITS(...). In this subparameter, you code the name of the exits that are to be invoked for the entire system or for certain subsystems.

Table 1 - Which SMF Exits Are Called for This Subsystem?

Exit Point	SUBSYS Value				
	JES2	JES3	STC	ASCH	TSO
IEFACTRT	Yes	Yes	Yes	Yes	Yes
IEFUAV	No	No	No	Yes	No
IEFUJI	Yes	Yes	Yes	Yes	Yes
IEFUJP	Yes(2)	No	Yes	No	No
IEFUJV	Yes	Yes	Yes	Yes(1)	Yes
IEFUSI	Yes	Yes	Yes	Yes	Yes
IEFUSO	Yes(2)	No	Yes	No	No
IEFUTL	Yes	Yes	Yes	Yes	Yes
IEFU29	No	No	Yes	No	No
IEFU29L	No	No	No	No	No
IEFU83	Yes	Yes	Yes	Yes	Yes
IEFU84	Yes	Yes	Yes	Yes	Yes
IEFU85	Yes	Yes	Yes	Yes	Yes

1. Notes:

IBM suggests that you use IEFUAV instead of IEFUJV to validate accounting information for APPC/MVS transaction programs. For more information, see [z/OS MVS Installation Exits](#).

2. The installation can cause this exit to be bypassed on a job class basis, through the JOBCLASS(v) initialization statement. For more information about the JOBCLASS(v) statement, see [z/OS JES2 Initialization and Tuning Reference](#).

Copyright IBM

User exits are usually written by the systems programmer, although some are provided as samples by IBM and others are offered by other users at no charge.

All of the SMF exits are shown in Table 1. They're also described in the section starting on page 14 where I describe the life of a job/TSO user/started task/etc.:

- IEFACRT - Accounting exit after step end and job end - This exit is typically used to display step and job totals on the JES log.
- IEFU29 - Called prior to dumping SMF from disk. This exit isn't discussed in this newsletter.
- IEFU29L - Called prior to dumping SMF from the SMF logger. This exit isn't discussed in this newsletter.
- IEFU83 - Called prior to writing SMF record - Can be used to modify a record or to keep it from being written. Several vendors use this exit to obtain SMF data needed for other applications.
- IEFU84 - Called prior to writing SMF record when branch-entered and not in cross-memory mode - Can be used like the IEFU83 exit.
- IEFU85 - Called prior to writing SMF record when branch-entered and in cross-memory mode. Can be used like the IEFU83 exit.
- IEFUJI - User job initiation - Often used to validate jobnames and accounting information.
- IEFUJP - User job purge - Used before the type 26 record is written before the job is purged from spool. I really don't know how people use this exit.
- IEFUJV/IEFUAV - User job validation and user APPC/MVS validation - The IEFUJV and IEFUAV exits can be used to edit and modify the JCL. The exits are actually called three times: before the line is processed by the converter, after the line is processed by the converter, and one time after all lines have been processed. These are the most common exits used to validate job names and accounting codes, and to enforce job classes.
- IEFUSI - User step initiation - This exit gets control before each step starts. This exit isn't used a lot, but could be used to validate the accounting codes at the step level or perform actions based on the time of day.
- IEFUSO - User sysout limit - Called when the job exceeds the spool output that is specified for the job class. The exit can extend the limit or cancel the step.
- IEFUTL - User time limit - Called when the job exceeds the CPU time limit that is specified for the job class. The exit can extend the limit or cancel the step.

You can specify which exits are to be invoked for all systems on the SYS parameter or by subsystem on the SUBSYS parameter of SMFPRMxx.

Performance Impact

Exits are primarily used to provide security, an audit trail, or additional reporting. For the added function, you would expect some overhead. The overhead directly depends on the programming ability and efficiency of the exit programmer. Some of the exits are called only a few hundred times a day (e.g. IEFUJV and IEFUJI are called once per job). Others are called several hundred thousand times a day. (E.g. IEFU83 is called for almost every SMF record.) The IEFU83, IEFU84, and IEFU85 exits should be very efficient!

The CPU time for executing exits will show up in various places in the job's CPU times. It might show up in the TCB time, the SRB time, or the initiator CPU time. An inefficient exit is difficult to identify because the CPU times are distributed across all of the jobs.

Recommendation

Your installation will determine which exits are required based on their chargeback, reporting, or audit needs. My primary suggestion is that you understand the reason for each exit and document it in parmlib member PROGxx, which defines all exits. You'll still need to maintain SMFPRMxx to indicate which exits are to be called for each subsystem.

If an exit is only needed for one subsystem, then include it on the SUBSYS statement, not the SYS statement. Watch out when IBM adds new exits. As an example, some people were using IEFU83 to intercept every SMF record before writing it. When IBM added IEFU84 (and later added IEFU85), most people didn't notice. If only IEFU83 is specified on the EXITS statement, then records written from branch-entry calls (and there are many) are not seen by the exit.

If you don't want to code your own exits, there are two software products that provide exits (and don't require Assembler knowledge). We mentioned these in our Tuning Letter 2006 No. 2 on page 34 (titled *Assembler Programmer Help*).

They are OS/EM from Trident Services

(www.triserv.com) and Easy/Exit from DTS Software (www.dtssoftware.com).

All SMF exits
have to be
very, very,
very efficient

Life of an Address Space

An address space may be one of four types of spaces according to SMF: TSO users, batch jobs, started tasks (including system address spaces), and APPC scheduler (ASCH) address spaces. With the exception of the system address spaces, the lives of the address spaces are recorded from the initiation to the termination of the address space. A type 30 SMF record is the most common record type associated with address spaces.

Figure 2 shows the life of an address space in SMF processing and this discussion will continue to refer to that figure. The left-most column indicates the point in the life of the address space, along with the records produced. The items indicated by a ▼ show some of the elapsed times that may be calculated from various timestamps. And the right column identifies SMF exits that are invoked. Although I use the term 'job' throughout this article, I use it to mean any type of address space including TSO users and started tasks.

JOB START UP

As you can see from Figure 2, a job is read into the system and the JCL is scanned and converted. During this time, the IEFUJV SMF exit is invoked but no SMF records are written. The IEFUJV (job validation) exit can be used to edit and modify the JCL. The IEFUJV exit is actually called three times: before the line is processed by the converter, after the line is processed by the converter, and one time after all lines have been processed. This is the most common exit used to validate job names and accounting codes.

JOB INITIATION

TSO users and started tasks (STCs) are then initiated immediately, while batch jobs wait for a JES initiator. When the address space is initiated, the IEFUJI (job initiation) exit is invoked and an SMF record is written. A type 30, subtype 1, record is written containing information from the job card, the job identification, and several timestamps. A job is uniquely (hopefully!) identified by the job name (or TSO userid or STC procedure name) and the reader timestamp (the time the job was first recognized by JES). Unfortunately, the timestamp is only precise to a hundredth of a second. As machines have gotten faster, this has proven to be a problem since it is possible for multiple jobs with the same job name to have the same reader timestamp! In a few instances, this has produced invalid results for these jobs. A solution to this has been to use the JES job number to further uniquely identify a job.

Just before the SMF record is written, the IEFU83, IEFU84, or IEFU85 exit is called.

The initiation record is a wonderful record just by itself. You can collect the following information from the type30, subtype 1 (written 30/1 from now on):

Job identification: Job name, reader timestamp

Job initiation timestamp

JCL parameters: Job class, accounting codes

Calculations:

Input queue time = initiation timestamp - reader timestamp

Figure 2 – The Life of an address space

JOBS SUBMITTED	
Reader active time	
CONVERTER STARTS	(IEFUJV)
Converter active time	(IEFUJV)
CONVERTER ENDS	(IEFUJV)
Input queue time	
JOB INITIATES (30/1)	(IEFUJI)
	(IEFU83/4/5)
Job creation time	
STEP 1 STARTS	(IEFUSI)
Dataset enqueue time	
Device allocation	
Device allocation time	
Problem pgm loaded	
Program execution time	(IEFUTL)
	(IEFUSO)
	(IEFU83/4/5)
INTERVAL ENDS (30/2)	(IEFACTRT)
STEP 1 ENDS (30/3 & 4)	(IEFU83/4/5)
STEP 2 STARTS	(IEFUSI)
Dataset enqueue time	
Device allocation	
Device allocation time	
Problem pgm loaded	
Program execution time	(IEFUTL)
	(IEFUSO)
	(IEFACTRT)
	(IEFU83/4/5)
Job termination time	
JOB ENDS (30/5)	(IEFACTRT)
	(IEFU83/4/5)
Output queue time	
OUTPUT WRITER STARTS	
Printer time	
OUTPUT WRITER STOPS (6)	(IEFU83/4/5)
Wait to purge time	
JOB PURGE (26)	(IEFUJP)
	(IEFU83/4/5)

Common plots:

- Jobs submitted per hour per class
- Jobs initiated per hour per class
- Average input queue time per hour per class

This information is used by:

- Service level management to evaluate input queue time.
- Performance analysts to help reduce input queue time.
- Capacity planners to determine volumes of job submissions by time of day and job class.
- Programmers to determine the best time of the day to submit jobs.

STEP INITIATION

Almost immediately after the job starts, the first step is initiated. The IEFUSI exit is called prior to beginning execution. No SMF records are written at this point.

STEP EXECUTION

As the step begins, there is a period of time spent enqueuing the datasets, and then the devices are allocated, after which the program is loaded and execution starts. These timestamps are kept for inclusion in later SMF records. During execution, if the amount of CPU time exceeds the time specified for the job class, the IEFUTL exit is called. This exit can be used to extend the time or to cancel the step. Also during execution, if the amount of lines written to spool exceed the number specified for the job class, the IEFUSO exit is called. This exit can be used to extend the number of lines or to cancel the step.

If INTERVAL was specified in SMFPRMxx for this type of address space, and the job is running longer than the interval, a type 30, subtype 2 record is written, after a call to the IEFU83/84/85 exit. In general, the data in the subtype 2 record contains the resources consumed for just the interval. The one exception is the type 30, subtype 6 record that is written for system address spaces at each interval. The subtype 6 records contain cumulative resource usage.

STEP TERMINATION

When a step completes, either normally or abnormally a type 30/4 is written. This record provides the majority of the data used for address space analysis. It contains total consumption for the major resources: CPU, I/O and storage. If INTERVAL recording was active, a type 30/3 is also written and contains the resource usage for the last (partial) interval. The IEFU83/84/85 exit is called before the SMF record is written. In addition, the IEFACTRT exit is called. This is one of the most popular exits and is used to place step termination information in the JES log. Those of you who see summary information of CPU usage and times at the end of every step probably have an IEFACTRT exit.

The step termination record – 30, subtype 4, is probably the most important SMF record

The major items collected in the 30/3 and 30/4 records include:

CPU measurements:

- TCB milliseconds
- SRB milliseconds
- Initiator TCB & SRB time
- Region control task (RCT) time
- Hiperspace CPU time
- Service units for all of the above
- Service definition coefficients
- zIIP and zAAP CPU times, with adjustment factors

I/O measurements:

- EXCPS per DD
- Connect time per DD
- BlockSize, device type, device unit

Storage measurements:

- Central storage frames
- Paging by private and common
- Swapping
- Virtual storage region

Additional information:

- Program name, step name, procedure name
- Completion code
- Active time
- Residency time
- Program start time

As you can see, lots of information is available at step termination. The following information is normally used by charge back, capacity planners, performance analysts and data center reporting:

Total CPU time in seconds or service units.

CPU time spent on zIIPs and zAAPs.

Total EXCPS or device connect time (note that SYSOUT usage cannot be provided at this point).

Number of TSO transactions or (TGETs and TPUTs).

Additionally, the following information is often gathered:

Program name and CPU time to determine users of software products.

Program name and other measurements to determine impact of software products.

Elapsed time to determine service level agreements.

Determination of who was using specific devices during a period of time.

Tape mount usage.

Device allocation time (program start time - allocation start).

JOB TERMINATION

Actually, the records created at job termination are very similar to those written at step termination. Many installations find that job totals are sufficient for chargeback and capacity planning. However, you should consider the instances where the job record does not get produced. For example, a job may have completed execution of ten steps during the last eight hours and the system crashes. You'll never see the job termination record but you probably want to account for the eight hours of usage, so the step termination records are important.

At job termination, the IEFACTRT exit is called again. The exit normally has logic to handle both step termination and job termination in slightly different ways. The exit normally produces output in the JES log showing job totals for resource usage, condition codes, and timestamps.

A type 30, subtype 5 is produced, with the IEFU83/84/85 exit being invoked before the record is written. Major differences between step termination records and job termination records include:

- CPU totals are job totals, not step totals.

- EXCP totals are job totals, not step totals and include LINKLIST activity, JES2 I/O (for JES), catalog management and OPEN/CLOSE.

- Job termination contains job card info such as account codes, programmer name field, job class, etc. while step termination doesn't.

After the job terminates, it will remain on the JES spool until all of the output is printed or purged. This could take days or even weeks.

SYSOUT MEASUREMENTS

A type 6 record is written for every SYSOUT dataset that is printed or routed to another location. The IEFU83/84/85 exit is invoked before the type 6 record is written. The type 6 provides line and page counts, as well as information on form usage. This information is used for charge back to recover printer costs; by capacity planners to determine print volumes by printer, location and form; and by service level management to determine when SYSOUT completed printing. Some studies can also be done on the length of time to print output.

There are five subtypes for the type 6 record depending on the type of output created:

- 0 - External Writer
- 2 - JES2 Output Writer
- 5 - JES3 Output Writer
- 7 - Print Services Facility (PSF) Output
- 9 - IP Printway Output

Routing information includes the output route code and the output form number. Several studies can be made on number of lines per form or output location. This can be done for daily or weekly totals, as well as hourly totals. One of the major pieces of information that is not available is when the SYSOUT was released from HOLD status.

The problem with service level management and report output is that the user could specify a held SYSOUT class, and not release or delete the dataset for days. If you are trying to provide batch turnaround (including printing) within one hour, you'll never make your objectives due to held SYSOUT. For that reason, most service levels today are from the time the job is read in until the job terminates (but not until the output has been printed). Another item to note with type 6 records is that they can be produced before the step terminates if FREE=CLOSE was specified on the JCL.

PURGE MEASUREMENTS

When all SYSOUT has been printed, routed or deleted, a type 26 record is written. The IEFUJP exit is invoked, and the IEFU83/84/85 exit is called before the record is written. I haven't seen much use of the IEFUJP exit, so if you have a good reason for one, please let me know.

TSO command
recording is
difficult but useful

Type 26 is a fantastic record. If it contained total CPU time and account code information, it would be almost perfect! The primary information in the type 26 is significant timestamps in the life of the job: reader start and stop, converter start and stop, execution start and stop and output start and stop. Most of service level management information can be obtained in this record alone.

Additionally, for systems with NJE, all system ids are provided here. You can tell that a job was read in on one system, converted on a second system, executed on a third system and printed on a fourth. When this information is used with the timestamps, you can also see the impact on multiple systems by time of day, job class and user.

Another nice feature of the type 26 records is the inclusion of total SYSOUT lines written to spool, along with estimated and actual SYSOUT byte counts. You can use this to let programmers know how close they are to their limits (before abending).

TSO COMMAND RECORDING

The intention of the type 32 record is to provide counts and, as an option, resource information on TSO commands. Even though it doesn't provide response time, the capability to analyze TSO usage by command is very useful. (TSO response time is best collected in the RMF type 72 record, which I'll cover in a future Tuning Letter.)

A type 32 record is written at TSO logoff or interval end containing information about each command. What isn't useful, however, is that called programs all fall under the "CALL" command and CLISTs fall under the "EXEC" command so that you can't tell information about specific programs or CLISTs. Another negative when using this record is that aliases aren't combined (that is, "SE and "SEND" are reported separately). If you want specific information on CLISTs, you could write a command processor that invokes the CLIST and track activity on the command processor.

The type 32 record can be a useful tool for special studies to help you determine which TSO commands should be placed in PLPA and which should be relegated to the LINKLIB libraries. If DETAIL is specified in SMFPRMxx, an additional 40 bytes of data

per command is added to the record to provide TCB, SRB milliseconds, TGETs, TPUTs, transactions, EXCP count and device connect time. This is most often used for special studies by the performance analysts (turn it on for only one week and study the results).

See our Tuning Letter of January 1991 on page 18 for how to collect more command information.

INTERVAL RECORDING

Interval recording for type 30 and type 32 records allows you to specify that records are to be written at specific intervals, such as every 30 minutes. A type 30/2 is written during every interval but the last, where a 30/3 is written. These records contain the same information as the step termination, type 30/4, but only contain totals for the interval. The primary purpose of interval recording is to provide statistics in case the system crashes during a step. For a batch job, this may not be too important, but for TSO it's critical since a TSO step is the entire logon session. If you do not use interval recording for TSO and you have an 'unscheduled IPL' in the middle of the day, you will have lost all SMF measurements for logged on TSO users during the period of time up to the crash.

(Note to capacity planners - you'll still get your performance group service units from RMF measurements, but no individual totals by TSO user or job). In a charge back environment or one where capacity planners are using SMF data, this could amount to a loss of a day's worth of information or billing data. Interval recording is also used where you want to collect information by job by time of day. For example, my CICS address space job totals may show that I had 10,000 EXCPs to device 174, but I may need to know whether those were spread throughout the day or during a single period of activity. Interval recording provides that information.

You can specify interval recording by subsystem (batch, TSO, STC, and APPC) in SMFPRMxx in SYSI.PARMLIB. Different interval times can be specified for each subsystem. An unfortunate side issue of intervals is the complexity of processing them. In general, they need to be sorted prior to processing. For example, I may only want to use interval records if I lost the step termination records. Since the 30/4 follows the 30/2 and 30/3 records, I can't process them until I've passed them. And I can't simply turn off step termination (30/4) since if a job executes in less than an interval of time, only a 30/4 will be created and no interval records are written. All in all, it's a messy job to process them! Several installations have written exits to bypass interval records on everything but important workloads (onlines, production batch and TSO).

Type 30, subtype 6 interval records are written for system address spaces if interval recording is specified for the STC subsystem. This is the only way to get information on these system address spaces. These interval records differ from the subtypes 2 and 3 because they contain cumulative information, not simply totals for the interval. If you do not collect this data, you could underestimate your capacity load by 20%.

References

I think that the most complete information on SMF comes from **Dr. Barry Merrill**. Barry developed MXG, a SAS-based system that initially read SMF records and created a performance database used by performance analysts and capacity planners. Since the first release of MXG in the early 1980s, MXG has expanded to be able to process hundreds of input sources. In 1984, Barry published the 1984 MXG Guide, followed in 1987 with the MXG Supplement. These became widely read because they were the first manuals to describe how to actually use the SMF data. They're still considered important references today.

Current SMF information is now available in MXG Newsletters, and often explained in the MXG-L ListServer. The newsletters, although written for MXG users, are accessible to the public. You don't need to license MXG to benefit from Barry's wisdom, but you'll soon want to have MXG for your own use. It's a great bargain, and I would never want to be without it. The website is www.mxg.com, and you can sign up for the listserver there.

IBM Manual, *z/OS V1R10.0 MVS Initialization & Tuning Reference*, [SA22-7592-17](#).

Describes parameters in SMFPRMxx parmlib member.

IBM Manual, *z/OS V1R10.0 MVS Installation Exits*, [SA22-7593-14](#). This manual describes how to use the SMF exits.

IBM Manual, *z/OS V1R10.0 MVS System Management Facilities (SMF)*, [SA22-7630-18](#).

This is the primary manual for SMF, and provides record layouts, and a description of the SMF macros and report programs.

Watson, Cheryl, DVD of Cheryl Watson's Tuning Letters 1991-2008. Because SMF is mentioned in almost every Tuning Letter, I can't really include a complete reference here. But the search on the DVD is effective, so if there's anything you want more information about, just use that tool to search the DVD. As an example, DDCONS brings up 50 references in 13 documents.

Watson, Cheryl, *SMF Update – Part 2*, Cheryl Watson's Tuning Letter 2008 No. 3, SMFPRMxx analysis. ■

z/OS 101

In this installment, I'll describe sources for SMF data regarding volume activity, file activity, and total I/O activity. Probably half of all I/O tuning is done at the volume level, and half is done at the data set level. The total I/O activity is generally used by capacity planners.

These records, unless noted otherwise, are described in the *IBM z/OS V1R10.0 MVS System Management Facilities (SMF)* manual, [SA22-7630-18](#). The service I'm providing here is to identify all of the sources of I/O information within the many, many SMF records.

Volume-related SMF Records

Volume, or device, statistics are used by installations for performance management, trending of space usage, and defining the I/O configuration (e.g. number of cylinders online). I use the terms volume and device interchangeably.

Volume Configuration

If you simply want to understand how much capacity you have, or how many volumes you have, you'll need to use several types of SMF records because the number of volumes can change throughout the day or since IPL. The change can come through an operator command or reply, or it can come from a dynamic reconfiguration. The following SMF records can help you:

- Type 8 - Configuration - The type 8 record is written at IPL, and contains the DASD or tape device number, device class and unit type of all devices that are online at IPL. This is used with the type 9 and type 11 records to provide a configuration picture.
- Type 9 - VARY Device ONLINE - A record is written every time a DASD or tape device is varied online and contains the same information as the type 8 record.
- Type 10 - Allocation Recovery - A record is written at the successful completion of device allocation and contains the same information as the type 8 record, along with the identification of the job that requested the allocation.
- Type 11 - Vary Device OFFLINE - A record is written every time a DASD or tape device is varied offline and contains the same information as the type 8 record.
- Type 19 - Direct Access Volume - The type 19 records are used to determine the amount of free space on each DASD volume. Unfortunately, until z/OS 1.10, they didn't include the amount of used space or the size of each volume. These records are used to track the amount of free space on all of the DASD volumes. One record is written for each DASD device at IPL and SMF switch. The data includes the device number, volume serial number, and the number of unused cylinders and tracks, as well as the largest contiguous space. In z/OS 1.10, it also provides the total amount of space in both cylinder-managed and track-managed areas. RMF Monitor III can provided volume usage and

free space with its SGSPACE parameter. You can obtain the data for space groups (SPACEG) or for devices (SPACED). This information is written to a VSAM database or simply displayed instead of recording to SMF.

- Type 21 - Error Statistics by Volume - This record is written every time a tape volume is unmounted from a device. It contains statistics for the duration of the mount time, including volume serial number, number of bytes read and written, number of I/Os, and number of errors on the tape. When tapes were less trustworthy, this was excellent information to help identify both volumes and tape drives that were getting dirty.
- Type 22 - Configuration - A type 22 record is written after every SMF initialization, after every operator CONFIG command, and when a dynamic configuration change is made with HCD. The only type of record that applies to volumes is the dynamic configuration change. This shows information about any device that was added, deleted, or modified. This does not contain any space information.

z/OS 1.10
now includes
device capacity in
Type 19 records!

Volume Activity

These SMF records show you the activity that occurs on each volume. Although I refer to many of the type 70 records as RMF records, these same records are also produced by BMC's CMF Monitor software product, which produces the same records as RMF, IBM's Resource Measurement Facility.

- Type 74 - RMF Activity (provides the following subtypes for volumes):
 - 1 - Device Activity - This is the most important record for analyzing device activity. One record is produced for each device at each RMF interval. In z/OS 1.10, the capacity of each device is also provided in the record, a wonderful addition!
 - 5 - Cache Subsystem Activity - One record is produced each interval for each cache control unit. It provides information to help tune the control unit by collecting read/write hit ratios, number of accesses, response time, etc. This is the primary record for tuning cache.
 - 7 - FICON Director Statistics - This record is used to identify both the configuration and the volume of activity for each FICON Director. This is also used for tuning the FICON paths and switches.
 - 8 - Enterprise Disk System Statistics - Provides performance and configuration information for Enterprise Disk Systems. These are produced at each RMF interval for each ESS (Enterprise Storage System).
- Type 78, subtype 3 - RMF I/O Queuing Activity - This record collects activity by logical control unit, and produces one record per interval containing all LCUs and PAVs. It gives the physical response times and queuing contention. It's used with the type 74, subtype 1 record to tune DASD volumes and PAVs.

- Type 79 - RMF Monitor II Activity - Provides Enqueue Contention (subtype 7), Device Activity (subtype 9), Channel Path Activity (subtype 12), and I/O Queuing (subtype 14). - This record provides the same type of information as the type 74 record, but at shorter intervals. Its online use allows you to select a device and then drill down to the jobs currently using the device. This is a very powerful tool. RMF Monitor III provides similar information but records that information to a VSAM database and not SMF.
- Type 94 - IBM Tape Library Dataserver Statistics - This record is written each hour and contains statistics about IBM's Tape Library Dataserver. The data is collected for all hosts. If you have a VTS (Virtual Tape Server) library, microcode level F/C 4001, and Outboard Policy Management is enabled, a subtype 2 record is also produced and contains Volume Pool Statistics. This record gives information about the number of drives, the avg/min/max mounts, the avg/min/max mount/demount times, compression ration, and import/export statistics. It's used for tuning and capacity planning.

File-related SMF Records

There are many SMF records produced to help you analyze file activity. Different SMF record types are written for each type of file.

Non-VSAM, non-zFS, non-HFS Data Set Activity

There are several SMF records that can help you analyze or track data set activity:

- Type 14 - Input Data Set Activity - When a non-VSAM data set used for input is closed, this record is written to provide the data set name, the number of I/Os, and the job step performing the close.
- Type 15 - Output (non-VSAM) Data Set Activity - The type 15 record provides the same type of data as the type 14 record, but is created for files that are opened as output or update.
- Type 17 - Scratch (non-VSAM) Data Set Status - A type 17 record is written whenever a non-VSAM data set is deleted. If a file is missing, you can use this record to determine which program or user deleted it. Be sure that you use either REC(PERM) or use the default instead of specifying REC(ALL) in SMFPRMxx, or else you will get a huge number of records for temporary data sets.
- Type 18 - Rename non-VSAM Data Set - A record type 18 is written whenever a non-VSAM data set is renamed. Again, this is used to locate a missing file.

VSAM Data Set Activity

There are even more SMF records that let you analyze VSAM catalogs and data sets:

- Type 36 - ICF Catalog - A type 36 record is written at the successful export of an Integrated Catalog Facility (ICF) catalog. It shows the date, time, and job that did the export.

- Type 41 - DIV Objects and VLF Statistics - Do you consider an I/O to storage an I/O? If so, then this record provides activity to in-storage objects, such as DIV (Data in Virtual) and VLF (Virtual Lookaside Facility). You can obtain the number of reads and writes, as well as the amount of storage used.
- Type 60 - VSAM Volume Data Set Updated - This record is written when the VVDS is updated. This can be used to help restore lost or damaged VVDSes.
- Type 61 - ICF Define Activity - A type 61 is created whenever a data set is added or updated in an ICF catalog. This contains a copy of the catalog record.
- Type 62 - VSAM Component or Cluster Opened - This is written for each open of a VSAM data set.
- Type 63 - VSAM Catalog Entry Defined - This record is obsolete because it applied to VSAM catalogs, which are no longer supported (as of January 2000).
- Type 64 - VSAM Component or Cluster Status - This is the most important record for VSAM files. One record is created when a VSAM file is closed, when it switches to another volume, or when it is out of space. It contains all of the information for the file during the time it was open, so shows the I/O activity and the size of the file. This is the record that is most often used to tune VSAM data sets.
- Type 65 - ICF Delete Activity - This record is written when a data set is deleted from an ICF catalog. This includes non-VSAM data sets that are scratched, and uncatalog requests even when the data set isn't deleted. This contains a copy of the catalog record.
- Type 66 - ICF Alter Activity - A record type 66 is written whenever an IDCAMS ALTER request (such as a rename) is made. This contains a copy of the catalog record.
- Type 67 - VSAM Catalog Entry Deleted - This record is also obsolete as of January 2000 because VSAM catalogs stopped being supported.
- Type 68 - VSAM Catalog Entry Renamed - This record is also obsolete as of January 2000.
- Type 69 - VSAM Data Space Defined, Extended, or Deleted - This record is obsolete as of January 2000.

Other Type of Files

SMF records are also produced for specialty files, such as HFS, zFS, and OAM.

- Type 42 - DFSMS Statistics and Configuration - This record produces several subtypes for SMS-managed volumes. Most of the information is used to help tune buffers and files.
 - 1 - Storage Class Summary & Buffer Management - Provides summary information by storage class. Written at timed intervals.
 - 2 - Cache Control Unit (3990-3) Statistics - Provides cache hit information. Written at timed intervals.
 - 3 - SMS Configuration Change - Identifies any changes to the SMS configuration due to a VARY SMS command, and ACTIVATE command, a SETSMS command, or a VARY command to an SMS-managed volume.

- 4 - System Data Mover (SDM) Statistics - Provides information on SDM activity, and is usually written at the end of the concurrent copy session.
 - 5 - Storage Class I/O Statistics - Contains storage class VTOC and VVDS I/O statistics, and written at the SMF global interval.
 - 6 - Data Set I/O Statistics - Contains data set I/O statistics and is written when the data set is closed, and also at the end of a type 30 interval (see previous Tuning Letter).
 - 7 - NFS File Timeout Statistics
 - 8 - NFS User Logout Statistics
 - 9 - Out of Space ABEND (Sx37) - Identifies jobs that abend due to out of space conditions.
 - 10 - Allocation Volume Selection Failure - Written when a job attempts to allocate a data set and there isn't enough space.
 - 11 - XRC Interval Statistics - Provides statistics for extended remote copy (XRC) sessions at the end of the SMF global interval.
 - 14 - ADSM Accounting - Provides accounting data for the ADSTAR Distributed Storage Manager (ADSM) when each session completes.
 - 15 - VSAM RLS Storage Class Statistics - Provides VSAM record-level sharing (RLS) storage class response time. Written at timed intervals.
 - 16 - VSAM RLS Data Set Statistics - Written at timed intervals to provide response times for RLS data sets.
 - 17 - VSAM RLS CF Lock Statistics - Contains data about the lock structure usage for the VSAM RLS coupling facility (CF), and is written at timed intervals.
 - 18 - VSAM RLS CF Usage Statistics - This record gives information on the RLS CF cache partition usage, and is written at timed intervals.
 - 19 - VSAM RLS Buffer LRU Statistics - Provides a summary of the VSAM RLS Local Buffer Manager LRU Statistics. Written at timed intervals.
 - 20 - All PDSE Members Deleted - Identifies a job or user who deletes all member from a PDSE file.
 - 21 - PDS/PDSE Member Deleted - Identifies a job or user who deletes a member of a PDS or PDSE file.
 - 22 - DFSMSrmm Audit Records - Contains rmm (removeable media manager - i.e. tape) audit records. This is new with z/OS 1.10.
 - 23 - DFSMSrmm Security Records - Contains rmm security records. This is new with z/OS 1.10.
 - 24 - PDS/PDSE Member Added or Replaced - Identifies a job or user who adds or replaces a member of a PDS or PDSE file. This is new with z/OS 1.10.
 - 25 - PDS/PDSE Member Renamed - Identifies a job or user who renames a member of a PDS or PDSE file. This is new with z/OS 1.10.
- Type 74 - RMF Activity, subtype 6 (HFS Statistics) - This record is written at the RMF timed intervals and contains information about the HFS Global Buffers, such as the size of the buffers and the hit ratio of the buffers. It also contains I/O and buffer statistics for each cataloged HFS data set.
 - Type 77 - Enqueue Activity - The reason I bring up this record is that I've used it to identify conflicts with data sets where jobs enqueue on the same data set. Performance can be improved if the contention is removed. There is one re-

cord produced per RMF interval containing all enqueues (including the data set enqueues) during the period.

- Type 85 - OAM Transaction Performance - This record provides statistics from the DFSMS Object Access Method. There are 37 subtypes which provides volume and response data to help in all aspects of OAM, such as tuning and capacity planning. These subtypes are documented in the DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support ([SC35-0426-06](#)).
- Type 92 - File System Activity - This record provides statistics on UNIX file systems and UNIX files, and is used for tuning the file systems. There are several subtypes:
 - 1 - File System Mounted - Provides File System name, time of mount, block size, total space, and used space.
 - 2 - File System Suspended or Quiesced - Provides File System name and time of suspend or quiesce.
 - 4 - File System Resumed or Unquiesced - Gives the File System name and time of resume or unquiesce.
 - 5 - File System Unmounted - Provides the File System name, the block size, total space, used space and activity since the mount (read/write calls, directory I/O blocks, I/O blocks read/written, bytes read/written), and time of unmount.
 - 6 - File System Remounted - Contains the same information as the subtype 5.
 - 7 - File System Moved - Contains the same information as the subtype 5, and includes the reason for the move.
 - 10 - File Opened - Provides time of open and the open file token & device number that can later be matched to the subtype 11 when closed.
 - 11 - File Closed - Provides the pathname of the file, the time of close, and the same activity statistics as the subtype 5.
 - 12 - Memory Map Started (MMAP) - Provides time of the mmap and a token to match to the subtype 13.
 - 13 - Memory Map Ended (MUNMAP) - Provides the time of the mmap, and the I/O blocks read & written.

Total I/O Activity

You can collect I/O totals by summing any of the volume or data set records. You can also obtain total I/O activity from another set of SMF records:

- Type 30, subtype 2, 3, 4, 5 - I/O sections - As we described in the last Tuning Letter, the type 30 records contains I/O sections for each address space termination or interval. The I/O section contains I/O counts by DDname in the subtype 2, 3, 4, and 5 records. Additionally, those subtypes contain fields for total I/O activity. So you can use this information to determine I/O activity by job step or TSO userid.

- Type 70 - RMF CPU Activity - This record contains the I/O interrupts per second by CPU. The total I/O interrupt rate of all CPUs multiplied by the number of seconds in the interval, provides the total I/O interrupts for the interval. Because each interrupt can process several returning I/Os, this number will not match any others, but it can provide tracking for long term.
- Type 72, subtype 1 - RMF Workload Activity - The RMF type 72 records contain totals by service class periods. These are shown in I/O service units per second. Division by the I/O service definition coefficient, multiplied by the interval, provides the total I/O count for the interval.

References

- BMC Software, [CMF Monitor](#). Manuals are available to BMC customers only.
- IBM Manual, *DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, [SC35-0426-06](#).
- IBM Manual, *RMF User's Guide V1R10*, [SC33-7990-14](#).
- IBM Manual, *z/OS V1R10.0 MVS Initialization & Tuning Reference*, [SA22-7592-17](#).
Describes parameters in SMFPRMxx parmlib member.
- IBM Manual, *z/OS V1R10.0 MVS Installation Exits*, [SA22-7593-14](#). This manual describes how to use the SMF exits.
- IBM Manual, *z/OS V1R10.0 MVS System Management Facilities (SMF)*, [SA22-7630-18](#).
This is the primary manual for SMF, and provides record layouts, and a description of the SMF macros and report programs.
- Merrill, Dr. Barry**, 1984 *MXG Guide*, and 1987 *MXG Supplement*. Available to MXG customers only, or purchasable from the MXG website at <http://www.mxg.com>.
- Merrill, Dr. Barry**, [MXG Newsletters](#).
- Watson, Cheryl**, *SMF Update – Parts 1, 2, & 3*, Cheryl Watson's Tuning Letter 2008 No. 2, 3, & 5. ■

System Logger

Starting with z/OS 1.9, it's possible for SMF to use the System Logger for storing SMF records. So this z/OS 101 installment will provide a basic understanding of the System Logger. I don't intend to cover everything you need to know to set up System Logger, because that's best covered by the wealth of IBM manuals, Redbooks, and white papers. But I do want to clarify some things. I'll be using the SMF logger for my data set name examples.

Terminology

IBM manuals, Redbooks, APARs, papers, and presentations seem to disagree about whether to use 'log stream' or 'logstream'. Even the Setting Up a Sysplex manual uses both terms, although 'log stream' is the more common usage in the manual. Because there seems to be no agreed upon term, I use the term 'logstream' because I prefer it. Be sure that you use both terms when making a search, such as for APARs. A recent APAR search found 264 hits for 'log stream', and 277 hits for 'logstream', although most are duplicates. Searching for 'logger' produced 413 APARs. A search on WSC found two documents for 'log stream', two different documents for 'logstream', and four documents (including a new one, but missing one of the others) for 'logger'. Be sure to try all combinations.

A big benefit is that you don't need to preformat logstreams

While I'm on the topic of terminology, be careful of how each document uses the word 'buffer'. In some documents, one buffer contains one block of SMF data; while in other documents, one buffer contains multiple blocks of data. I've used the second definition for this newsletter, but it can be confusing when you look at parameters. The BUFSIZMAX in SMFPRMxx means the maximum amount of space to be used for getmains of 8M buffers, each of which can contain multiple blocks of SMF records. But MAXBUFSIZE (with an 'E' at the end) in the LOGR policy definition of a logstream means the maximum size of one block of SMF records.

System Logger References

Unless otherwise specified, the links to manuals are for z/OS 1.10.

IBM Redbook, [System Programmer's Guide to z/OS System Logger, SG24-6898](#)

IBM, [MVS Setting up a Sysplex, SA22-7625](#)

Enrico, Peter, [System Logger — Understanding, Measuring, and Tuning](#), SHARE 2009 Proceedings, Session 2533

System Logger

The System Logger (abbreviated as Logger from here on) was introduced with Parallel Sysplex over 15 years ago. It provides a common method for logging and retrieving data, and can be used by many applications at the same time. There is an address

space, IXGLOGR, that runs in each MVS image. Its function is to manage all of the blocks of data from all of the applications using Logger.

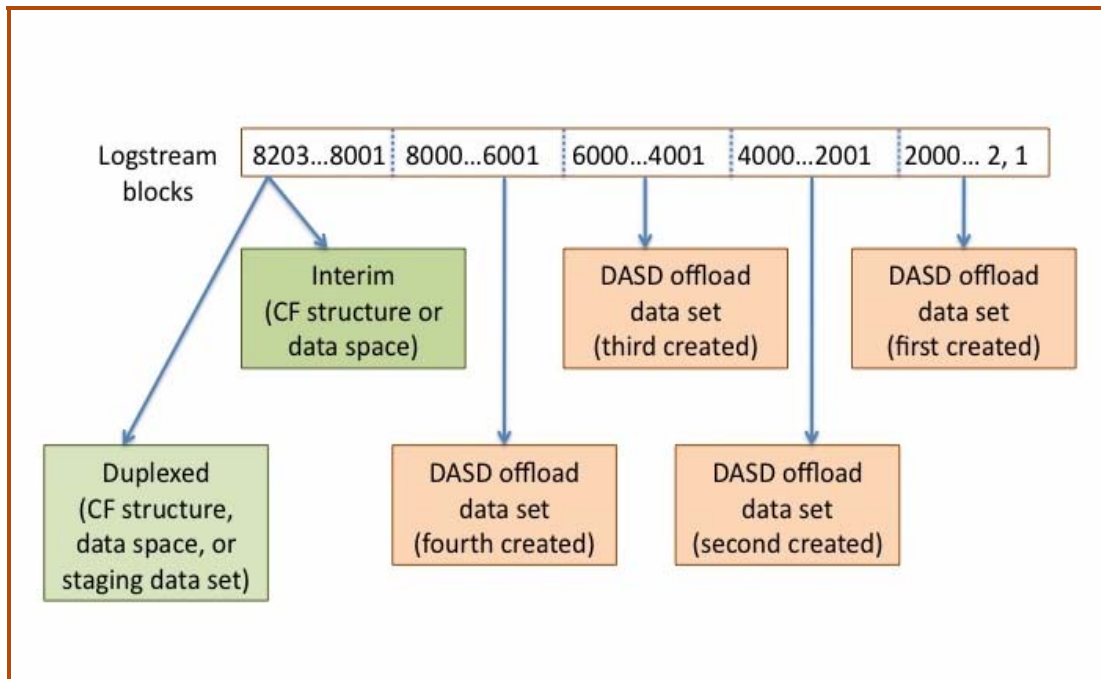
A single sysplex-wide LOGR couple data set (CDS) contains the policies for all exploiters of the System Logger, and includes a directory for every logstream showing where each piece of the logstream is located. A program that is accessing the logstream can ask to read it from oldest to newest, from newest to oldest, by a specific block number, or by a date and time.

In addition to managing logstreams, the Logger also write SMF type 88 records, which can be used to tune logstreams. I'll cover the type 88 record in another newsletter.

Logstreams

Blocks of data from each application are stored in logstreams. Logger adds a block number and a timestamp to the front of every block passed to it. A logstream is a logical, possibly very long, stream of blocks where the most recent blocks are kept in interim storage, and the older blocks are contained in DASD offload data sets. Interim storage can be a coupling facility (CF) structure or a data space owned by IXGLOGR. As the interim storage fills up, IXGLOGR will write the blocks out to DASD offload data sets. Logger will perform duplexing for each type of logstream. Optionally the user can specify additional, more secure duplexing (I won't be addressing optional duplexing options in this issue). Figure 3 shows the location of portions of a logstream.

Figure 3 - Location of Logstream Blocks



There are two types of logstreams that can be defined to the Logger — CF logstreams and DASD-only logstreams. Figure 4 contains a list of the attributes for each type of logstream. (The data set names I've used are for SMF — each application has its own naming convention.) A CF logstream is usually used when you want multiple MVS systems to record to the same logstream. For example, you might want to keep all SMF audit records from all systems in a single data set. In that case, you could create a CF logstream for just the audit records. Logger automatically duplexes a CF logstream to a data space owned by IXGLOGR. A DASD-only logstream may only be written to by one system. Logger automatically duplexes a DASD-only logstream to staging data sets. These are temporary data sets that are deleted when all of the interim data has been written to an offload data set, and the application has disconnected from Logger.

Figure 4 - Attributes of Logstream Types

Attribute	CF Logstream	DASD-only logstream
Interim storage	CF structure	IXGLOGR data space
Primary storage	DASD offload data set - hlq.IFASMF.name.Annnnnnnn	DASD offload data set - hlq.IFASMF.name.Annnnnnnn
Duplexing storage	Default - IXGLOGR data space; optional - a staging data set — hlq.IFASMF.name.sysplexname, or another CF structure	Always - staging data set - hlq.IFASMF.name.systemname
Size of SMF buffer	MAXBUFSIZE on structure definition	MAXBUFSIZE on logstream definition
OFFLOAD based on	CF structure percent full	Staging data set percent full

Defining Logstreams

Figure 5 shows an example of how to define a DASD-only logstream to the LOGR policy. You do not need to allocate or preformat these data sets. System Logger will allocate and create them as needed. The data set names for SMF must contain the node of 'IFASMF'. Logger will add a unique identifier (an 'A' followed by a 7-digit sequential number, starting at A0000001) to the end of the name on DASD. The parameters in this example are described in the Setting Up a Sysplex manual. I've highlighted the three parameters that I'll be referring to in the description of the offload logic. The LS_SIZE is the number of 4K blocks that define the maximum size of each DASD offload data set. For an SMF logstream, an installation could choose to make each offload data set big enough to hold 24 hours of data, or big enough to hold four hours of data. Please see the Setting Up a Sysplex manual for information on the other parameters.

CF logstreams must be defined first to the coupling facility resource management (CFRM) policy and then to the LOGR CDS with utility IXCMIAPU. You can specify one or more structures, with each structure containing one or more logstreams. See Figure 6 for an example of the CFRM policy and Figure 7 for an example of the LOGR policy.

Figure 5 - Example of JCL to Create a DASD-only Logstream

```
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
  DATA TYPE(LOGR) REPORT(YES)

  DEFINE LOGSTREAM
    NAME(IFASMF.SYS1.PRIMARY)
    DASDONLY(YES)
    MAXBUFSIZE(65532)
    STG_SIZE(10000)
    STG_DATACLAS(MVSLOGR)
    LS_SIZE(10000)
    LS_DATACLAS(MVSLOGR)
    HLQ(LOGGER)
    HIGHOFFLOAD(60)
    LOWOFFLOAD(35)
    AUTODELETE(YES)
    RETPD(1)

/*
```

Figure 6 - Example of Job to Define CF Structure in CFRM Policy

```
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(CFRM)
  . . .
  STRUCTURE
    NAME(LS_SMF_SYSPLEX)
    SIZE(10000)

/*
```

Figure 7 - Example of Job to Define CF Structure Logstream in LOGR Policy

```
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *

  DATA TYPE(LOGR) REPORT(YES)

  DEFINE STRUCTNAME(LS_SMF_SYSPLEX)
    MAXBUFSIZE(65532)
    LOGSNUM(1)

  DEFINE LOGSTREAM
    NAME(IFASMF.SYSPLEX.COMMON)
    STRUCTNAME(LS_SMF_SYSPLEX)
    LS_SIZE(10000)
    LS_DATACLAS(MVSLOGR)
    HIGHOFFLOAD(60)
    LOWOFFLOAD(35)
    AUTODELETE(YES)
    RETPD(1)
    OFFLOADRECALL(NO)

/*
```

Offloading Logstreams

The HIGHOFFLOAD and LOWOFFLOAD parameters in the LOGR policy define the frequency of offloads. For a CF logstream, when the CF logstream structure gets filled to the percent specified in HIGHOFFLOAD (60% in our example), all of the MVS systems are notified, and they will 'race' to perform the offload. There is no way to control or predict which system will do the offload. The IXGLOGR address space on the 'winning' system will then move blocks from the CF structure to the associated DASD offload data set. After the write completes, the blocks are then removed from the CF structure and the IXGLOGR data space (and any duplexing areas). For a DASD-only logstream, when the staging data set becomes 60% full, then blocks of data are moved from the data space to the DASD offload data set. After the write completes, the blocks are then removed from the data space and the staging data set (and any duplexing areas). Logstream blocks are moved to the offload data set until the percent of the interim storage in use is down to the LOWOFFLOAD value (35% in our case).

The size of the DASD offload data set is determined by the LS_SIZE parameter in the LOGR policy for each logstream. If the offload data set fills up to that maximum size, then a new data set with the next higher sequential number is created.

The retention periods and automatic deletions are only looked at during an offload activity, and no other time.

High values of HIGHOFFLOAD can be a problem if the data is being passed to Logger faster than Logger can start offloading the blocks. This could occur due to a high spike of unexpected records. The closer the range of highoffload and lowoffload, the more frequently Logger will offload the data. ■

The contents of this document are excerpted from

Cheryl Watson's
Tuning Letter

Published 6 times a year
by Watson & Walker, Inc.

<http://www.watsonwalker.com>

Publisher: Tom Walker
Executive Editor: Cheryl Watson
General Manager: Linda May

2009-2010 SUBSCRIPTION RATES: - Electronic version (DVD & email) \$950 per year (single-site license), including DVD of issues since 1991. Multi-site discounts are available. Subscribe online at: www.watsonwalker.com.

Payment may be made by a check drawn on a U.S. bank or any major credit card. Send all correspondence to: Watson & Walker, Inc., 7618 Sandalwood Way, Sarasota, Florida 34231, USA. Tel: 800-553-4562 or 941-924-6565. Fax: 941-924-4892. For customer service, send email to admin@watsonwalker.com. For technical questions, send email to technical@watsonwalker.com.

© 2009 Watson & Walker, Inc. All rights reserved. ISSN #1079-6606. Reproduction of this document is permitted only for internal use at the physical address where it was received. Permission is required for exceptions to this rule.

All IBM and other product names are trademarks of their respective owners.

Note: Implementation of any suggestions contained in this newsletter should be preceded by a controlled test and is the responsibility of the reader.