SMF: Old dog learning new tricks

Steven Jones

SMF/Scheduler/Allocation team lead
sbj@us.ibm.com

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

BigInsights	DFSMSdss	FICON*	IMS	RACF*	System z10*	zEnterprise*
BlueMix	DFSMShsm	GDPS*	Language Environment*	Rational*	Tivoli*	z/OS*
CICS*	DFSORT	HyperSwap	MQSeries*	Redbooks*	UrbanCode	zSecure
COGNOS*	DS6000*	IBM*	Parallel Sysplex*	REXX	WebSphere*	z Systems
DB2*	DS8000*	IBM (logo)*	PartnerWorld*	SmartCloud*	z13	z/VM*
DFSMSdfp						

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.
* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

1. **Using SMF data from TSO REXX**
2. **New SMF function in z/OS V2.1, V2.2, and Post GA V2.2**
3. **SMF Real Time Interface**
   A. Overview
   B. Setup and Configuration
   C. Overview of new callable services
      1. IFAMQRY
      2. IFAMCON
      3. IFAMGET
      4. IFAMDSC

Creating REXX exec to display SMF records

- I needed to see GRS records of ENQs/DEQs
  - Type 87, Subtype 2
  - Each record contains *up to* 30 entries for requesters
  - Mapped separately with equal number of entries for actual resource major/minor name
- IDCAMS prints unformatted data, but hard to match requesters to resources in array format
- Wanted a 1 request per line format for ISPF FIND/EXCLUDE function, 120 columns max.
- Wanted to see timestamp in GRS data, not SMF record time (needed STCKE formatting)

Creating REXX exec to display SMF records

- **Found a series of 4 articles in IBM Systems Magazine that detail how to**
  - read SMF records from IFASMFDP output
  - Extract individual fields and format as decimal, hex, EBCDIC, bit
  - (Added my own “format a timestamp” routine)
  - Write records to a different file
  - Even included downloadable JCL and REXX code for SMF Type 19!
  - http://www.ibmsystemsmag.com/mainframe/tipstechniques/applicationdevelopment/rexx_smf_part1/

Creating REXX exec to display SMF records

- Best way to start, copy existing EXEC and JCL
  - My EXEC and JCL are available from the NaSPA download site
  - Provided as-is, caveat emptor, YMMV, etc.
- In Process_SMF_Data routine, replace the SMF87 fields with calls to format routines for your record fields.
  - Example:

```
SMF87REQ_JOB=ebcdic(line,SMF87DEF_Req_Off+Offset(36),8)
SMF87REQ_TCB=binary_x(line,SMF87DEF_Req_Off+Offset(32),4)
OUTLINE.x = SMF87REQ_JOB||' '||SMF87REQ_TCB
```

  - Note: Be sure to use the OFFSET function in the EXEC. It accounts for EXECIO removing the vlen length.
- The format (offset/length) of fields in most records is described in the SMF book. GRS' SMF 87 is in the *Planning: Global Resource Serialization* book.

Creating REXX exec to display SMF records

- **Functions included in downloaded code (and GRS87RPT)**

```
Function: Binary_d              Returns : Decimal
Function: Binary4_d             Returns : Decimal--> for negative
                                    values in 4 Byte binary fields
Function: Binary_h              Returns : Decimal (same as binary_d)
Function: Binary_x              Returns : Hexadecimal
Function: Binary_b              Returns : Binary
Function: Packed                Returns : same as binary_x
Function: EBCDIC                Returns : EBCDIC
Function: Smftime               Returns : hh:mm:ss
Function: Smfjdate              Returns : Julian date yyyy.ddd
Function: TodDateTime           Returns : 27 char Human readable
                                    date/time
```

Creating REXX exec to display SMF records

- **Part of my formatted data (all from 1 SMF record):**

```
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelExc 2 006 00 SYSZTIOT 0067007E3FB0
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelShr 2 006 01 SYSZTIOT 0067007E3FB0
ObtShr 3 041 04 SYSDSN   DSNDSMX.DSNDBC.PEST24D.PEST24S.I0001.D667
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelExc 2 006 00 SYSZTIOT 0067007E3FB0
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelShr 2 006 01 SYSZTIOT 0067007E3FB0
ObtShr 3 041 04 SYSDSN   DSNDSMX.DSNDBC.PEST25D.PEST25S.I0001.D626
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelExc 2 006 00 SYSZTIOT 0067007E3FB0
ObtShr 2 006 00 SYSZTIOT 0067007E3FB0
RelShr 2 006 01 SYSZTIOT 0067007E3FB0
ObtShr 3 041 04 SYSDSN   DSNDSMX.DSNDBC.PEST24D.PEST24S.I0001.D668
```

SMF and z/OS V2.1

- **Job Wait Time enhancement**
  - SMFPRMxx includes parameters for many system functions
    - JWT(Job Wait Time) in SMFPRMxx allows SysProg to specify maximum amount of time that a job or TSO/E user address space is allowed to wait continuously
    - Specified as 4 digits: hhmm
    - New in z/OS 2.1: Separating out other types of work
      - TWT: TSO Wait Time

        TWT(0100) – Cancel TSO user if no activity for 90 seconds
      - SWT: STC Wait Time

        SWT(1000) – Cancel Started Task if no activity for 10 hours
    - Value here is to be able to set TSO and STC values different than user batch values
    - As with JWT, SMF exit IEFUTL will get control to decide whether to extend the wait time or Abend the unit of work.

SMF and z/OS V2.1 - Authsetsmf

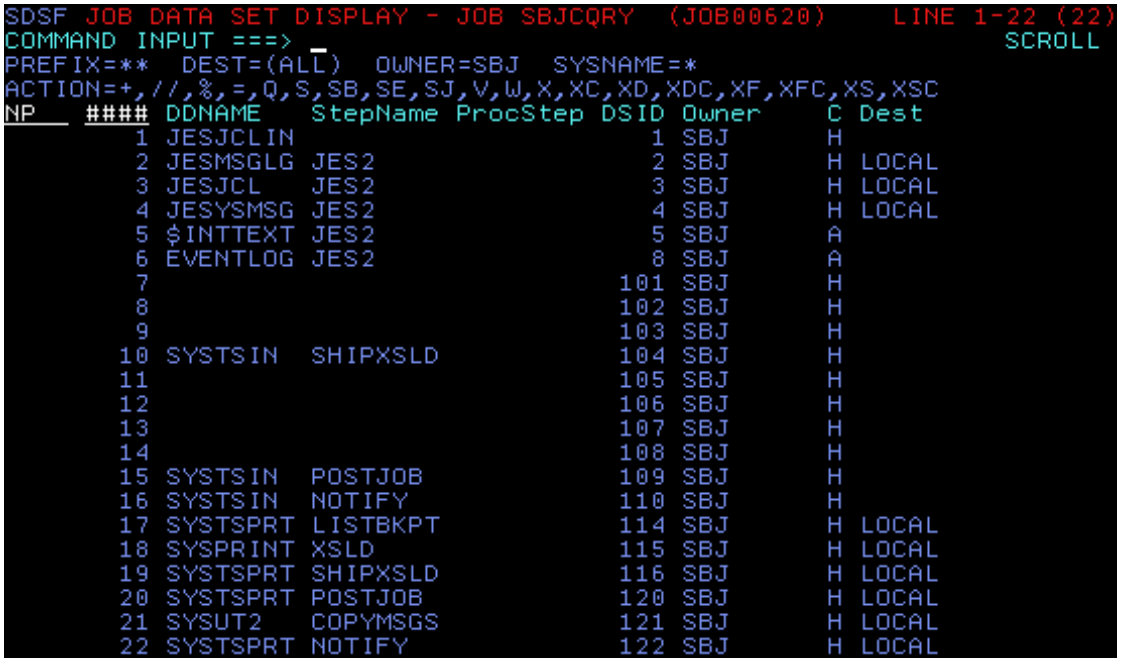
- **Allowing SETSMF command without prompting during IPL**
  - SMFPRMxx includes parameters for many system functions
  - PROMPT allows operator to inspect and alter any SMF parameter during SMF INIT
  - Problem: WTOR will delay IPL completion, so most use NOPROMPT
  - By default, NOPROMPT also says “Do not allow SETSMF command”, potentially restricting operator actions for configuration and error recovery
  - New in z/OS 2.1 – AUTHSETSMF/NOAUTHSETSMF
    - Default: NOAUTHSETSMF
    - AUTHSETSMF – Allow SETSMF even when “NOPROMPT” specified

SMF and z/OS 2.1 – SMFPRMxx COMPRESS

- **SMF introduced "Record to Logstream" support back in z/OS 1.9**
  - Much faster way for SMF to write records
  - Some customers still pushing the limits, esp. when requesting DB2 records
- **zEDC exploitation allows SMF records to be compressed before writing to System Logger:** Less data to store, faster writes of smaller data
- **Simple exploitation for people with zEDC already set up:**
  - Add "COMPRESS" to LSNAME parameter
  - May want to use PERMFIX parameter to ensure enough fixed storage available to write at speed
  - When dumping records from logstream, IFASMFDL automatically decompresses data
- **For best performance, need to ensure data can be decompressed by z/OS V2 system with a zEDC available**
  - SOFTINFLATE available, but can't match performance of hardware compression
- **Since z/OS V2.1, the following additional support has been added**
  - IFASEXIT support in OA49157
- **Redbook available:**
  - http://www.redbooks.ibm.com/redbooks/pdfs/sg248259.pdf

SMF and z/OS V2.2 – SMF Type 30 in job data

JES2 capturing SMF TYPE 30 records with other JOB data



- Jobs run on JES2 with z/OS V2.2 now include EVENTLOG spool data set
- Behavior can be disabled with $T JOBDEF,SUP_EVENTLOG_SMF=YES
- Among other records, it contains JOB START, STEP END and JOB END subtype records of SMF TYPE 30.
- Now, SMF data for Type 30 records is kept with other JOB data
- Can be viewed from SDSF
  - Requires INPUT ON
  - Use “?” line command at SDSF STATUS display
- Printing data from SDSF
  - Use SET PRTCCASA OFF to ensure no extra chars added
  - “PRINT” to a data set
- See z/OS JES Application Programming for EVENTLOG details https://www.ibm.com/support/knowledgecenter/SSLTBW_2.2.0/com.ibm.zos.v2r2.hasc300/accevnt.htm

SMF and z/OS 2.2 – IEFUSI replacement

- **SMF Exit IEFUSI is used by virtually all customers to set limits on storage**
- **IEFUSI has many inputs and outputs and must be coded in Assembler**
  - High cost for making updates, including for new workloads
- **z/OS 2.2 includes support for a new SMFLIMxx parmlib member**
  - Customers define a set of rules about a jobstep's REGION and storage LIMIT
  - Each rule contains
    - zero or more "filter" keywords (to match the rule to the step that is about to start)
    - One or more attribute fields to override what the IEFUSI exit (if any) returns
  - Each rule can override all or part of earlier rules that matched
    - Result is a compendium of the matching rules' attributes
  - Operator command support to SET SMFLIM and DISPLAY the current rules in use
  - JOB support (MSGIEF043I) to indicate applied limits for each job step.

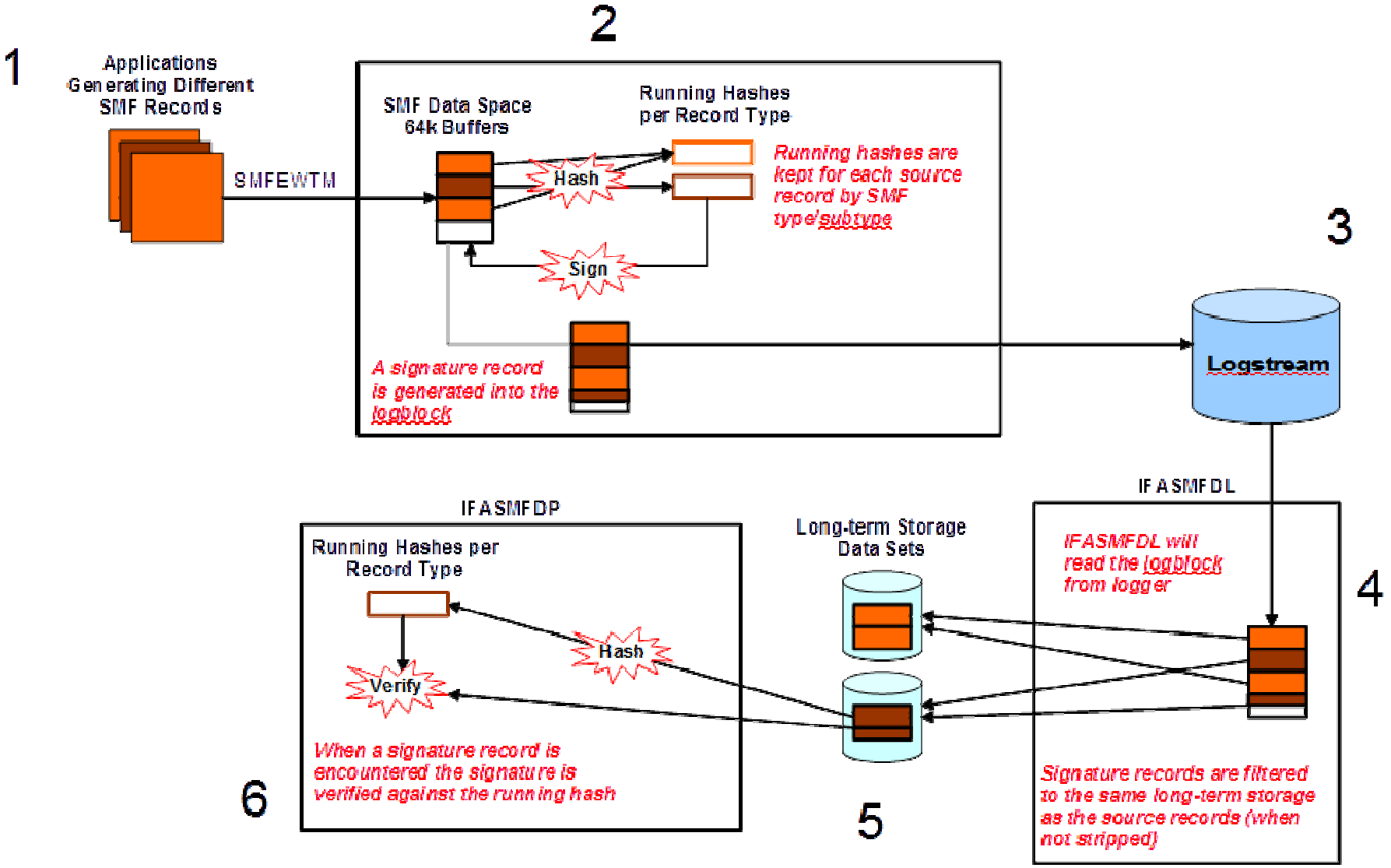
SMF and z/OS 2.2 – IEFUSI replacement

- **SMFLIMxx syntax**
  REGION <filter keywords> <attribute keywords>
  – Filter keywords (each with wildcard support):
    - JOBNAME,STEPNAME,PGMNAME (including “fetchlib”),
    - SUBSYS, JOBCLASS, USER,
    - JOBACCT, STEPACCT – matches any number of accounting data subfields
    - SYSNAME – to support using one parmlib member across multiple systems
  – Attribute keywords:
    - SYSRESVABOVE/SYSRESVBELOW – reserves high private storage for system uses – corresponds to LIMIT output parameters from IEFUSI
    - REGIONABOVE/REGIONBELOW – sets the REGION
    - MEMLIMIT – override MEMLIMIT set by JCL or SMFPRMxx MEMLIMIT
- **Designed to allow you to leave IEFUSI in place and override specific cases**
- **While the number of inputs is large, many customers can replace their IEFUSI with a simple SMFLIMxx:**
  REGION JOBNAME(*) SYSRESVABOVE(100M) SYSRESVBELOW(512K)
  REGIONBELOW(NOLIMIT) REGIONABOVE(NOLIMIT)
  – Preserves storage for required system function, allows everything else to be used

SMF and z/OS 2.2 – Tamper resistant SMF

- **SMF records contain critical information about an enterprise and are archived for long durations.**
- **The SMF records are generally shared among various departments for a number of activities.**
- **There is no built in protection of the SMF data**
- **Enter SMF Digital Signatures!**
- **SMF will hash the records, encrypt the hash and store with the record data to allow later verification of data via IFASMFDP**
  - No records altered
  - No extra records added
  - No records deleted.

SMF and z/OS 2.2 – Tamper resistant SMF



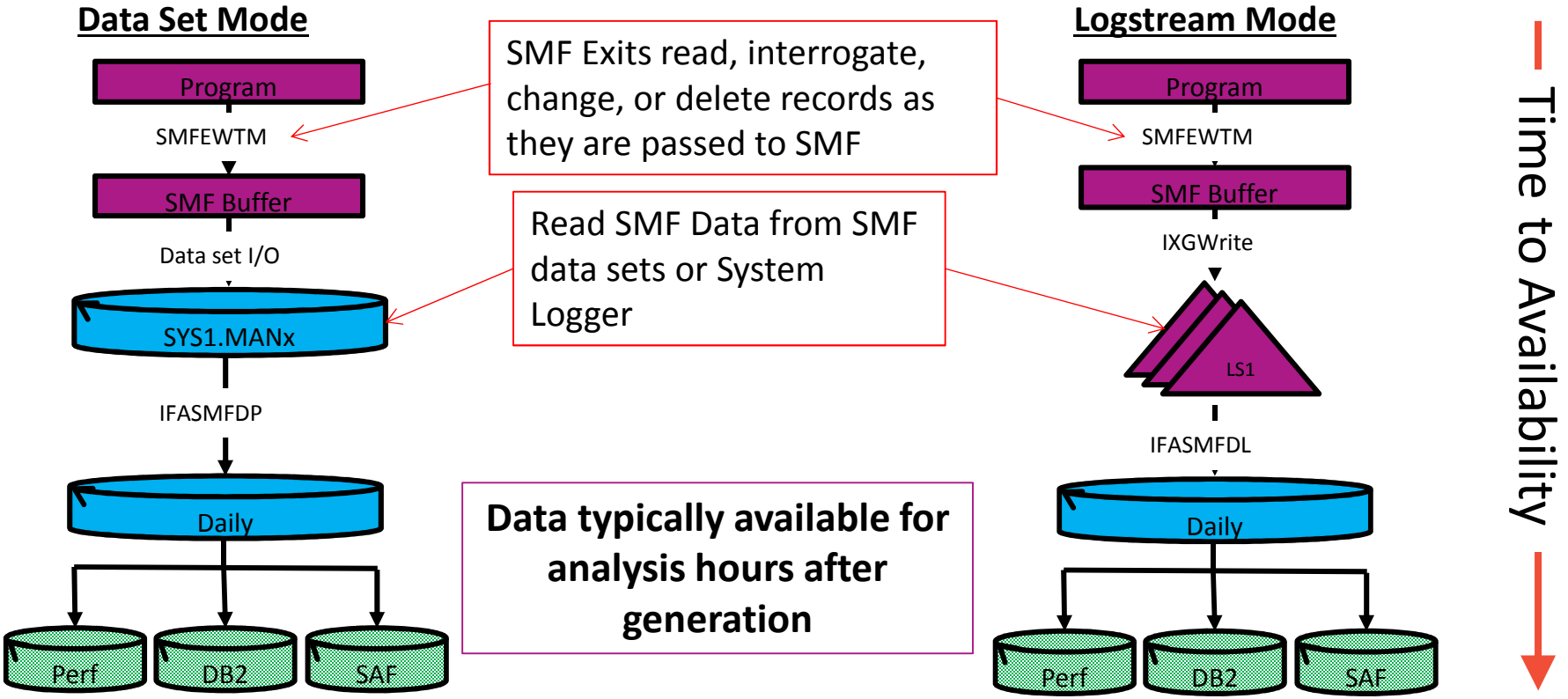
SMF and z/OS 2.2 – Tamper resistant SMF

- **Now, SMF records can be validated for audits, etc.**
- **Read more about it:**
  http://www.redbooks.ibm.com/iea/v2r2/pdf/zOS_V2R2_BCP_Tamper_Resistant_SMF.pdf

SMF and z/OS 2.2 (Post GA) – Real Time Interface

- **SMF data has always been intended to be stored in data sets and used for post-processing jobs**
- **Getting the data into a running program before it reached its final resting place was difficult**
  - Some people have “grabbed” the data as it passed through the IEFU8x exits
  - Some have even read directly from MANx data sets or System Logger

Current SMF Infrastructure



z/OS SMF Real-time Services

- **New Real-Time SMF services provided on top of existing buffer technology**
- **Define new "In-memory (INMEM) Resources" for specific records/sets of records**
  - Separate data space buffer for each resource, collecting records for that resource
  - Records are buffered until buffer wraps and overwrites older records
- **Can write any set of SMF records to a in-memory resource only – No disk required**
  - Co-exists with current SMF logstream technology
  - Not supported when RECORDING(DATASET) used
- **APIs allow application to access SMF data as it is buffered**
  - Unauthorized access policed via SAF
  - Connect/Get/Disconnect model similar to traditional QSAM access
- **Potential use-cases include**
  - Detecting security violations in real-time
  - Real time monitoring resource usage
  - Dynamic Job Scheduling based on current resource consumption

IEFU8x vs New Interfaces

Function	IEFU8X Exit	New RealTime Interface
Ability to Edit Records	Yes – Direct access to SMF record is provided	No – Record is copied to user's buffer
Minimum Authorization Requirement	Authorized load module, runs authorized	Problem State, Any Key
Caller's mode	Depends on the exit, IEFU83, IEFU84 and IEFU85 each provide access to different calling environments	P=H=S, Task mode
Record Selection	None – All records provided to the exit	Only records recorded to specific in-memory resource provided to caller
Security	Exit is authorized and has access to all records	Caller has to pass SAF check for access to that specific in-memory resource
Access Point	During SMFEWTM processing	After SMFEWTM processing has completed and returned to the caller

Setup and Configuration

- Up to 32 in-memory resources can be defined
  - Each resource can have a unique set of records collected to be read
  - Up to 8 programs can connect to resource to read records
  - Resource name similar to LSNAME – IFASMF.xxxx – up to 26 characters
- RESSIZMAX can be used to define the buffer size
  - Note: In memory buffer treated as a wrapping buffer that is never emptied
- Accepts all expected TYPE or NOTYPE statements
  - The TYPEs recorded in-memory will not be processed by DEFAULTLSNAME processing
    - Allows for InMemory only records (i.e. never written to permanent storage)

```
/*********************************************************************/
/*                                                                   */
/*  Define in-memory resources                                       */
/*                                                                   */
/*********************************************************************/
INMEM(IFASMF.INMEM.RES5,TYPE(0:30),RESSIZMAX(128M))
INMEM(IFASMF.INMEM.RES6,TYPE(30:90),RESSIZMAX(128M))
INMEM(IFASMF.INMEM.RES7,TYPE(0:127),RESSIZMAX(128M))
```

Setup and Configuration

- **SAF definition** – Do this before updating SMFPRMxx
  - Define a SAF resource for each planned INMEM resource
    RDEFINE FACILITY IFA.IFASMF.INMEM.RES1
    RDEFINE FACILITY IFA.IFASMF.INMEM.RES2
    RDEFINE FACILITY IFA.IFASMF.INMEM.RES3
    RDEFINE FACILITY IFA.IFASMF.INMEM.RES4
  - Permit userids to facility resources, e.g.
    PERMIT IFA.IFASMF.INMEM.RES1 CLASS(FACILITY)
    ID(zzzzzzzz) ACCESS(READ)
    … and so on
  - SETROPTS RACLIST(FACILITY) REFRESH

Setup and Configuration

- **SMF definition**
  - Ensure you start with a working SMFPRMxx member with RECORDING(LOGSTREAM)
  - Examine the TYPE/NOTYPE statements in SYS statement
  - Add any new TYPEs needed for In-Memory resources
    - Note that INMEM TYPEs may change usage of “DEFAULTLSNAME”
  - Add an INMEM statement for each resource
    - Must include TYPE (or NOTYPE) and RESSIZMAX subkeywords to be valid
    - Define the record types needed for the INMEM resource

      Ex. INMEM(IFASMF.INMEM.RES5,TYPE(0:30),RESSIZMAX(128M))
      INMEM(IFASMF.INMEM.RES6,TYPE(30:90),RESSIZMAX(128M))
      INMEM(IFASMF.INMEM.RES7,TYPE(0:127),RESSIZMAX(128M))
    - Expect all storage in RESSIZMAX to be used (over time)
  - Use SET SMF=xx to activate the new parmlib member

Programming Interfaces

- **IFAZSYSP**
  - Assembler mappings of input parameter blocks for:
    - IFAMCON – ConParmBlock
    - IFAMDSC - DscParmBlock
    - IFAMGET - GetParmBlock
    - IFAMQRY - QryParmBlock

      plus mapping of IFAMQRY output, QrPb_InMemResource
- **IFARCINM**
  - Constants for return codes and reason codes

Programming Interfaces

- **Query the names and record types of INMEM resources *to which the user has SAF access***
  - Call IFAMQRY with Reg1 pointing to 3 DWORD parmlist:
    - 64-bit addr of input structure (See QryParmBlock in IFAZSYSP)
    - 64-bit addr of 4 byte retcode
    - 64-bit addr of 4 byte rsncode
  - AMODE 64, task mode, unlocked
  - Output: Resource names and record types returned in array of QrPb_InMemResource entries in buffer, QrPb_ReturnedImrs tells how many

```
! Pseudo-code example
Set QrPb_Eyecatcher = QRPB#Catcher
Set QrPb_Length     = QRPB#ParmLen
Set QrPb_Version    = QRPB#CurVer
Set QrPb_BufferSize = <Your INMEM name, char 7-26>
Set QrPb_Buffer@    = <addr of your buffer for returned data>

<Use call logic similar to IFAMCON
Call IFAMQRY with parmlist pointing to QryParmBlock, retcode, rsncode
Check retcode. If successful, use QrPb_ReturnedImrs and QrPb_InMemResource
```

Programming Interfaces

- **Connect to an INMEM resource**
  - Call IFAMCON with Reg1 pointing to 3 DWORD parmlist:
    - 64-bt addr of input structure (See ConParmBlock in IFAZSYSP)
    - 64-bit addr of 4 byte retcode
    - 64-bit addr of 4 byte rsncode
  - AMODE 64, task mode, unlocked
  - Output: Connect token returned in ConParmBlock
    - Token is unique to the address space and can only be used by the ASID that made the connection

```
! Pseudo-code example
Set CnPb_Eyecatcher = CNPB#Catcher
Set CnPb_Length     = CNPB#ParmLen
Set CnPb_Version    = CNPB#CurVer
Set CnPb_Name       = <Your INMEM name, char 7-26>
Set CnPb_NameLength = <length of INMEM name>

Define a V-CON for the IFAMCON
Set AMODE 64 if needed
Call IFAMCON with parmlist pointing to ConParmBlock, retcode, rsncode
Check retcode. If successful, save CnPb_Token
```

Programming Interfaces

- **GET a record**
  - Call IFAMGET with Reg1 pointing to 3 DWORD parmlist:
    - 64-bit addr of input structure (See GetParmBlock in IFAZSYSP)
    - 64-bit addr of 4 byte retcode
    - 64-bit addr of 4 byte rsncode
  - AMODE 64, task mode or SRB mode, unlocked
  - Output: SMF record in buffer pointed to by GtPb_Buffer@

Continued, next slide

Programming Interfaces

- **GET a record,** continued

```
! Pseudo-code example
Set GtPb_Eyecatcher = GtPB#Catcher
Set GtPb_Length     = GtPB#ParmLen
Set GtPb_Version    = GtPB#CurVer
Set GtPb_Token      = <Token saved from Connect>
Set GtPb_NonBlocking (if do not want a “blocking GET”)
Set GtPb_BufferSize = <size of buffer for next record>
Set GtPb_Buffer@    = <addr of your buffer for returned data>

<Use call logic similar to IFAMCON>
Call IFAMGET with parmlist pointing to GTPB, retcode, rsncode
Check retcode. If successful, use record in buffer. Gtpb_ReturnedLength
indicates length of data returned
```

Programming Interfaces

- **Disconnect from an INMEM resource**
  - Call IFAMDSC with Reg1 pointing to 3 DWORD parmlist:
    - 64-bt addr of input structure (See DscParmBlock in IFAZSYSP)
    - 64-bit addr of 4 byte retcode
    - 64-bit addr of 4 byte rsncode
  - AMODE 64, task mode, unlocked
  - Output: return/reason code only

```
! Pseudo-code example
Set DsPb_Eyecatcher = DSPB#Catcher
Set DsPb_Length     = DSPB#ParmLen
Set DsPb_Version    = DSPB#CurVer
Set DsPb_Token      = <Token saved from Connect>

<Use call logic similar to IFAMCON>
Call IFAMCON with parmlist pointing to DsPB, retcode, rsncode
Check retcode. If successful, save DsPb_Token
```

Hints and Tips

- **Use of types with INMEM may alter what is received by DEFAULTLSNAME**
- **If using the "blocking GET",**
  - Be prepared for return codes that indicate that your job was cancelled, that SMF was forced, or that a SET SMF= command is removing your resource.
  - Be prepared for a resource buffer wrap where they could lose some records and be re-syncronized to the newest record in the dataspace (both in blocking and non-blocking mode).
- **Expected flow of service calls:**
  1. Call IFAMQRY to get INMEM names and record types defined
     - Locate a resource or validate resource has correct set of records being collected for its purpose
  2. Call IFAIMCON to connect to the resource
  3. Do while no failures
     1. Call IFAIMGET to get a record from SMF

        Assume GET is blocked (waiting) until a record available
     2. On return, process the record
     3. Check for external condition to exit. If exit needed, exit loop
  4. After loop, call IFAIMDSC to disconnect
  5. Exit

In-memory resources and Real Time Interface

- **Available now via APAR OA49263**
- **PDF doc available:**

  http://publibz.boulder.ibm.com/zoslib/pdf/OA49263.pdf
- **Questions?**
- **Suggestions for new function?**
  https://www.ibm.com/developerworks/rfe/execute?use_case=submitRfe

  Brand: Servers and Systems Software

  Product Family: z Systems Software

  Product: z/OS

  Component: BCP_SMF