SMF Real Time Interface

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Installation
- Session Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - “None”.

Session Objectives

- Understand the the purpose of the SMF Real-Time function
  - Advantages using SMF Real-Time instead of existing exits
  - Overview of the function and API
- Understand the SMF changes
  - SYS1.PARMLIB updates
  - SMF Initialization updates
  - Overview of API
  - Usage

Overview

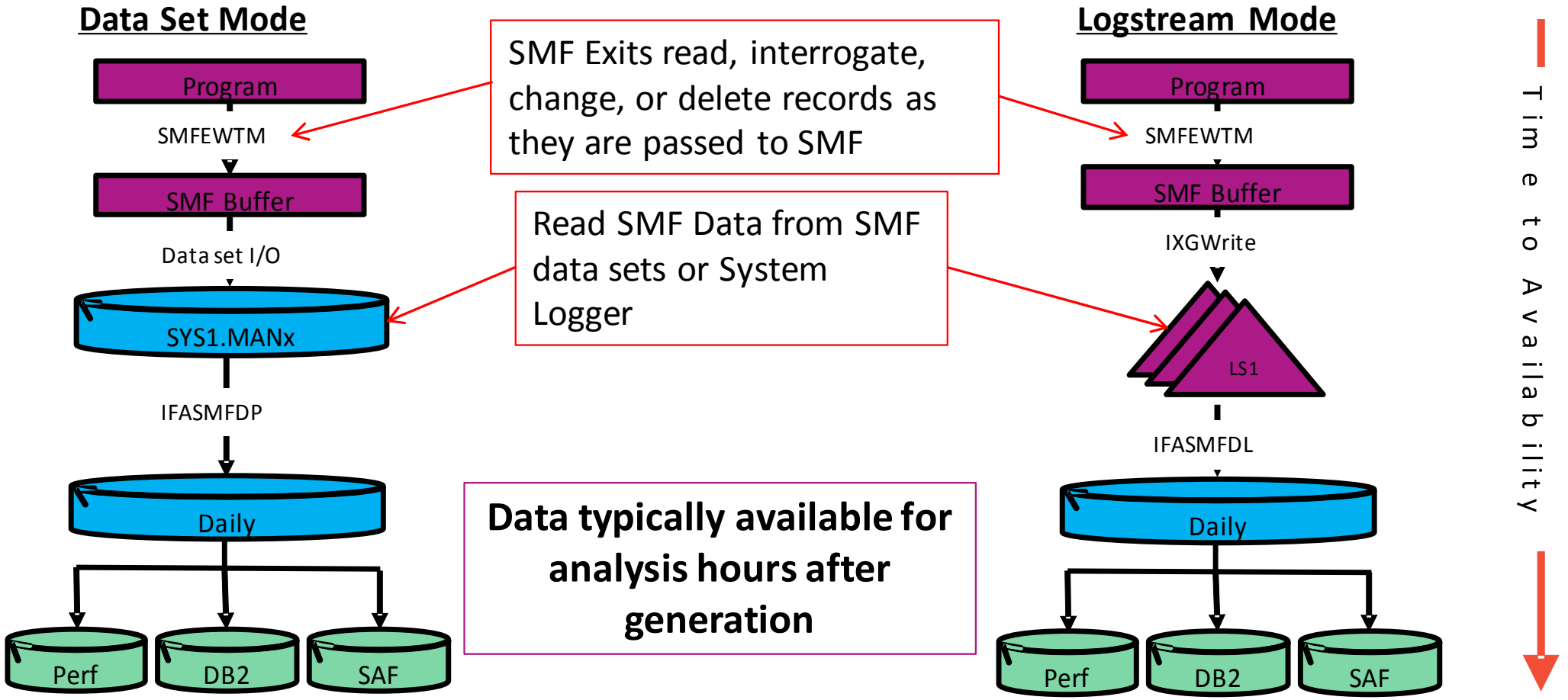
- Problem Statement / Need Addressed
  - Many applications and system components generate SMF records that contain valuable information that are valuable for IT analytics
    - Today, those SMF records are buffered and eventually written to either a logstream or SYS1.MANx data sets
    - Records are available for analytics processing later
      - Can be minutes to several hours after the events occurred
    - A possible solution is to use the IEFU83/IEFU84/IEFU85 exits to get access to records as they are written, but this has several drawbacks
      - Requirements on user exit code
        - Maintain low latency as exit is called before record is buffered
      - No ability to provide a subscription for specific SMF types
        - All records are provided to the exit.
      - This is an authorized code path
        - Limits the ability of client applications to access the data without operational concerns.

Overview

- Solution
  - Provide a new SMF Real-Time API for applications
    - Not a replacement for existing SMF recording
    - An alternative that provides near real-time SMF records for monitoring and data analysis

- Benefit / Value
  - New INMEM configuration that is easy to implement
    - Leverages existing logstream data buffering code and also the capability to segregate records into different target buffers
    - Non-disruptive to SMF processing
    - Access to in-memory resources is protected via SAF

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
  - 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

Overview

IEFU8x Exits vs SMF Real-Time API

Function	IEFU8X Exit	New Interface
Ability to Edit Records	Yes – Direct access to SMF record is provided	No – Record is copied to user's buffer after any modification by IEFU8x exits
Authorization Requirement	Authorized load module, runs authorized	Supervisor or Problem State, Any Key
Caller's mode	Depends on the exit, IEFU83, IEFU84 and IEFU85 each provide access to different calling environments	P=H, Task mode for connecting, disconnecting, query SRB mode allowed for GET requests
Record Selection	None – All records provided to the exit	Only records recorded to specific in-memory resource provided to caller (as specified in SMFPRMxx)
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 In-Memory only records (i.e. never written to permanent storage)

```
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
  - 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 - Macros

- **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 - API

- Four callable services interfaces
  - IFAMCON – Connect to a named INMEM resource and obtain token
  - IFAMDSC – Disconnect from an INMEM resource using token
  - IFAMGET – Get one or more records from the INMEM resource associated with a connect token
    - Supports blocking and non-blocking reads
    - New function to support reading multiple records to be available soon (OA51878 - March 2017)
  - IFAMQRY – Query INMEM resources
    - Only returns resources that your program is authorized to read
- Services support AMODE 64 and data areas in 64 bit storage
- Single system in nature
  - Records from other systems not available in INMEM resource

Programming Interfaces – Hints and Tips

- **If using the “blocking GET”,**
  - Be prepared for return codes that indicate that your job was canceled, that SMF was forced, or that a SET SMF= command is removing your resource.
  - Be prepared for a resource buffer wrap when records are written faster than the application reads them
    - The application could lose some records
    - The application will be re-synchronized to the newest record in the data space (both in blocking and non-blocking mode).

Programming Interfaces – Hints and Tips

- **One suggested 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 IFAMCON to connect to the resource
3) Do while no failures
   1) Call IFAMGET 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 IFAMDSC to disconnect
5) Exit

Interactions & Dependencies

- Software Dependencies
  - Z/OS V2.1 or z/OS V2.2
    - Install PTFs for OA49263 and OA51878
  - Function included in z/OS V2.3 base
- Hardware Dependencies
  - None
- Exploiters
  - Rocket Software
  - IBM Tivoli
  - IBM zSecure
  - SPARK

Migration & Coexistence Considerations

- With regard to DEFAULTLSNAME
  - In order to support sending a record to INMEM only, a side effect is that records written to INMEM DO have a valid “target” and are not written to a DEFAULTLSNAME
  - This means that records that are deliberately sent to a DEFAULTLSNAME logstream will be directed to the INMEM resource instead, resulting in them not being permanently recorded

SMF Display Commands

- DISPLAY SMF
  - Under the status, “IN-MEMORY” will show for in-memory resources

```
IFA714I 08.21.08 SMF STATUS 031
          LOGSTREAM NAME                BUFFERS         STATUS
         A-IFASMF.INMEM30               74641508        IN-MEMORY
```

- DISPLAY SMF,**M**
  - Shows the in-memory resources and if any active connections

```
IFA714I 08.28.37 SMF STATUS 052
         IN MEMORY CONNECTIONS
           Resource: IFASMF.INMEM30
           No Connections
           Resource: IFASMF.INMEM
           Con#: 0001 Connect Time: 2016.242 08:28:31
           ASID: 003E
           Resource: IFASMF.INMEM127
           No Connections
```

SMF Message changes

- New message
- `IFA847I DUPLICATE INMEM in-memory_resource_name REJECTED`
  - **Explanation:** During SMF initialization or during SET SMF=xx or SETSMF command processing, the system issues this message to indicate that an in-memory resource name that is defined for SMF data in the SMFPRMxx parmlib member was previously specified in the parmlib member.
  - In the message text:
    - *in-memory_resource_name*
      The incorrect in-memory resource name. If the original in-memory resource name in the parmlib member specified a symbol, such as &SYSNAME. or &SID., this message will display the in-memory resource name after the value of the symbol has been substituted.

Installation

- To exploit INMEM resources:
  - Authorization updates required
  - SMFPRMxx updates required

Session Summary

- With the SMF Real-Time Interface, your programs can see SMF records created in near real time
  - To monitor system utilization externalized via SMF data, “as it happens”
  - To collect and analyze data for decision making

Appendix

- Publications:
  - z/OS Introduction and Release Guide (GA32-0887)
  - z/OS MVS System Management Facilities (SMF) (SA38-0667)
  - z/OS MVS Initialization and Tuning Reference (SA23-1380)
  - z/OS MVS System Commands (SA38-0666)
  - z/OS MVS Programming: Callable Services for High-Level Languages (SA23-1377)
  - z/OS MVS System Messages, Vol 8 (IEF-IGD) (SA38-0675)
  - z/OS MVS System Codes (SA38-0665)
- z/OS 2.2 Pub updates available at :
  - http://publibz.boulder.ibm.com/zoslib/pdf/OA49263.pdf