

Spark and SMF Fasten Your Seatbelt, This is Going To Be A Wild Ride!

Session 525

Frank Kyne, Watson & Walker

Frank@watsonwalker.com



Welcome



- Thank you for attending this session. If you love SMF data, I think you are in for some interesting times ahead.
- Who am I.
- Cheryl says 'Hi!'
- Thanks!
 - To all the people in Rocket Software and IBM that helped me get up and running with Spark.
 - To YOU for attending.
- These slides are VERY different from those on your memory sticks – if you would like the latest version, please email me (frank@watsonwalker.com).
- Disclaimer – no financial interests in IBM or Rocket, generally no idea what I'm talking about. Anything useful you pick up is an unintended bonus...
- If you have questions, please ask as I go along.

What are we going to talk about?



- A journey, starting 40+ years ago, with a destination in the future.
- A little background on SMF processing – why it worked the way it did, what challenges that created, and how IBM is changing it to position it for the future.
- What exactly is Spark?
- And what does it have to do with my SMF data?
- A real world sample of how it could be used.
- Considerations/Concerns.
- The future.
- Summary.
- Detailed implementation steps to get you started (for your reference)

SMF History



- For those that are not familiar with SMF, what is it?
 - Callable system service to save information to a central repository in a standardized format.
 - Some of the data is created by the operating system. The service is also used by transaction managers, database managers, performance tools, queue managers, and many other IBM *and* non-IBM products.
 - Some examples of the types of data it contains are:
 - Chargeback, security violations and auditing, disk space usage, system performance, WebSphere reporting, software usage (what runs where), CPU usage, processor cache and memory usage, file-level performance, processor capacity and usage, transaction performance and resource usage, database request performance and resource usage, network usage, many many others.
 - The description of the fields for the IBM SMF records takes 810 pages in the [SMF manual](#). And that doesn't even include the non-IBM products (and even some IBM ones!).

SMF History

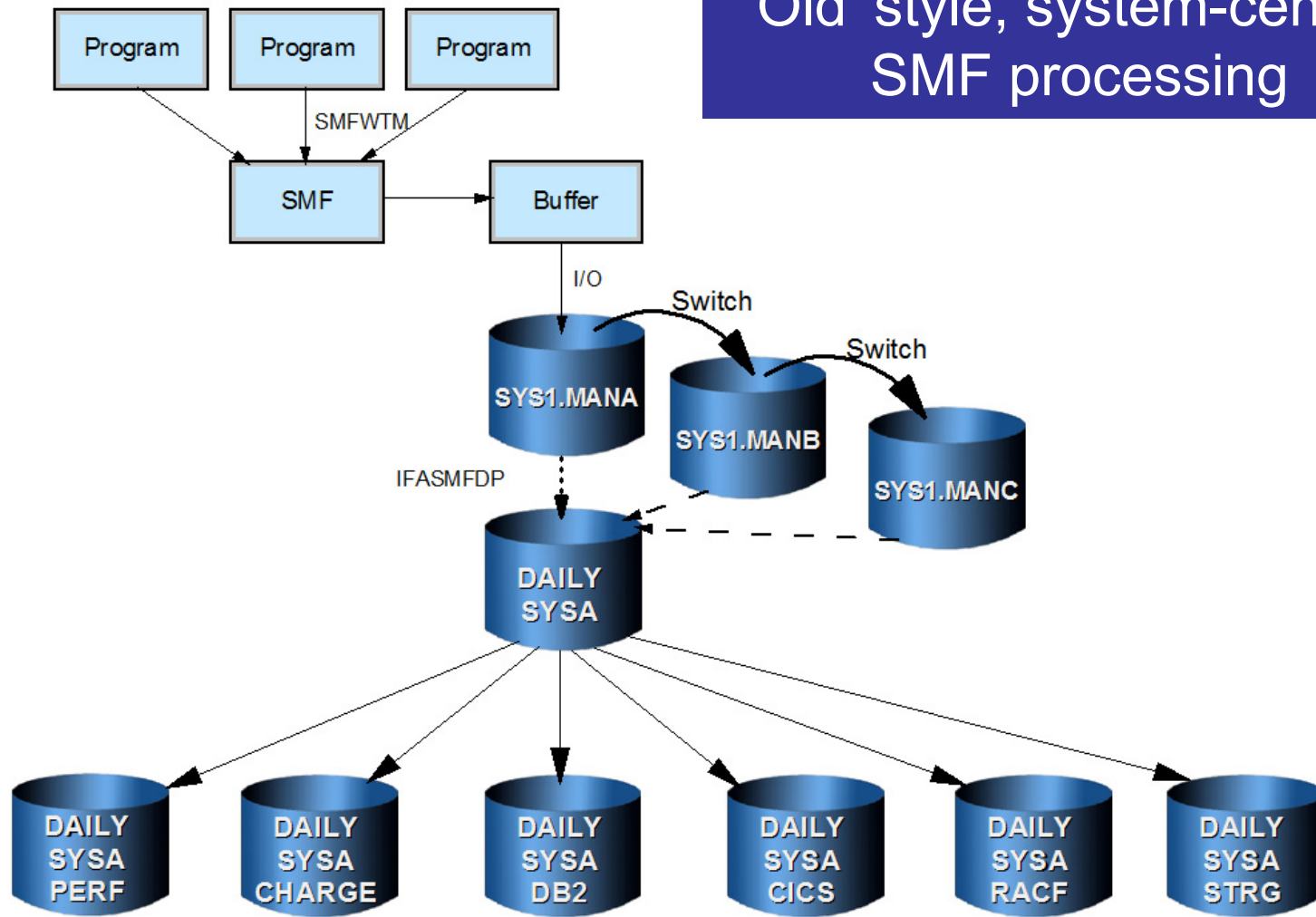


- SMF was invented back in the day when most customers had only 1 system.
- Originally, ALL SMF records for a system were written to one repository. With only 10 or 20 record types at the time, and no concept of data sharing or workload balancing, this made sense.
- Because of the wealth of information the SMF records contained, an industry of products that processed various SMF record types quickly built up. The most well known is probably SAS and Barry Merrill's MXG. But there are many other products that target specific niches (some of those vendors are here at the conference).
 - Because of the structure of SMF records, processing them is not trivial, especially with products that expect data to be in fixed length records with fixed offsets.

SMF History



'Old' style, system-centric, SMF processing



Time=0

Time=+1440

SMF Challenges

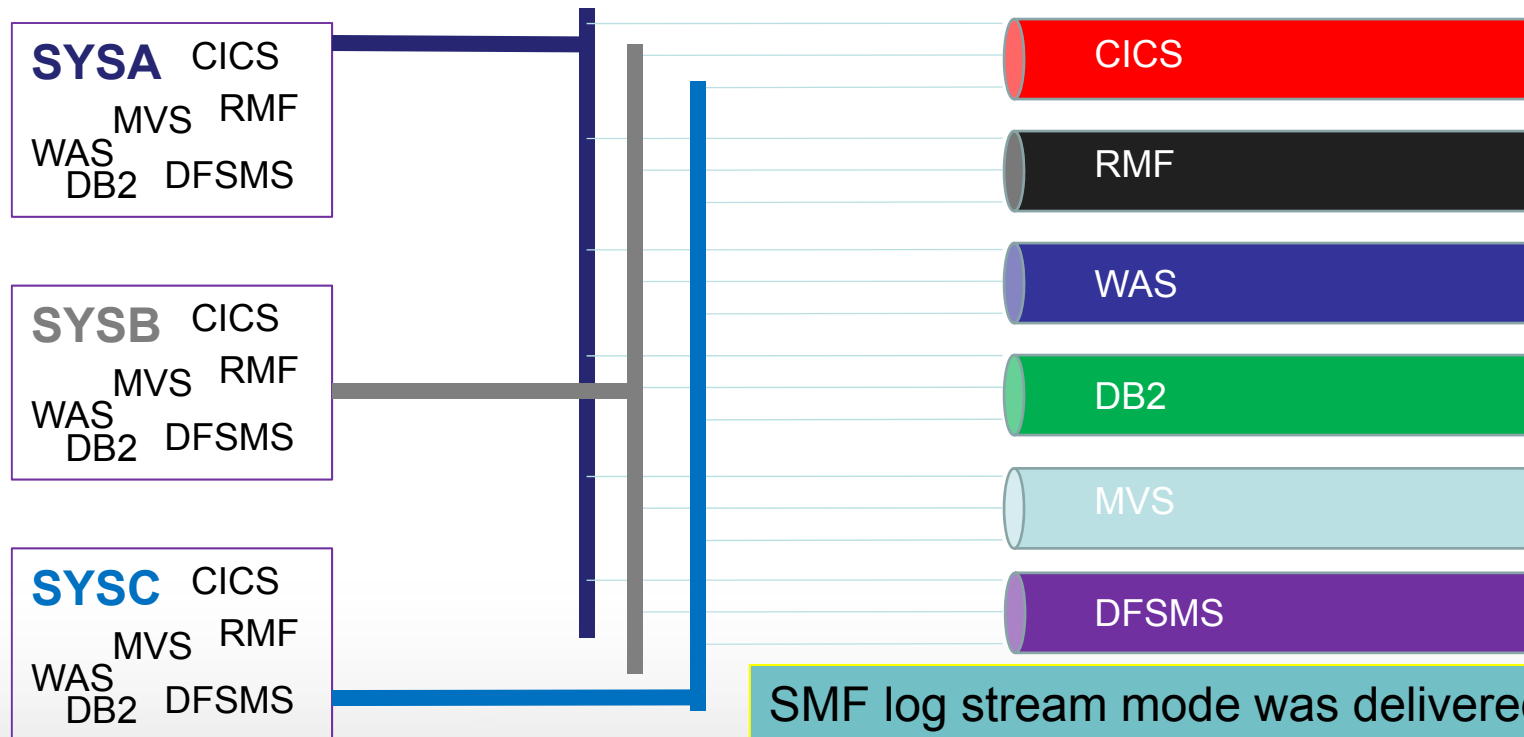


- As time went by, the size and number of systems grew, sysplex provided the concept of single system image (so applications were spread over multiple replicated systems), and the departments that supported those systems grew and became more specialized (Perf, Stor Mgmt, Database, etc).
- All of this resulted in:
 - More products that *exploited* the SMF capability.
 - MUCH more data (I know of one customer that creates > 2.6TB of SMF data *per day* (that's equivalent to 390 *strings* of 3330s per day!), for just one sysplex).
 - Lost records because records are created faster than they could be written to the SYS1.MAN data sets.
 - A desire to use features such as striping or compression to enable higher volumes. Old style VSAM used for SYS1.MAN data sets does not support any of these features.
 - A need to handle SMF data at the function level (CICS records for all systems) rather than at the system level (all record types for one system).

SMF Log Stream mode



- IBM's response to these requirements was to add log stream support to SMF.



SMF log stream mode was delivered in 2007. So it will no longer be 'new' next year, so therefore it is safe to consider it. 😊

SMF Log Stream mode



- The introduction of log stream support for SMF had the potential to:
 - Let customers structure SMF repositories based on how they *use* the data.
 - Support vastly increased volumes of SMF data (log streams are nearly infinitely scalable and flexible).
 - Nearly eliminate the risk of losing SMF records as a result of SMF buffers filling.
 - Provide much higher levels of resiliency and security for the SMF records.
- And what did all this mean? Yet MORE SMF data..
 - Sites could now enable SMF record types that were previously impractical because of bandwidth limitations on the SYS1.MAN data sets.
 - Yet more record types and subtypes – that is, products are saving more information into SMF.

Tips for logstreams:
If you use SMF Logstreams, make sure you use ARCHIVE rather than DUMP followed by DELETE.
And use SMARTENDPOINT and SMARTEPOV keywords in your IFASMF DL jobs.

Recent Enhancements to SMF



- Using the SMF log stream support as a base, IBM have since enhanced SMF further by adding support for zEDC compression.
 - This typically achieves compression ratios of between 8 and 10 to one for SMF data.
 - You can compress both the logstream (including the data in the offload data sets) and in the sequential data sets if you archive the SMF records from the log stream, saving significant amounts of disk space.
- To further protect the integrity of the information in the SMF records, IBM introduced SMF record signing in z/OS 2.2.
 - Like zEDC, this is also only available if SMF is in log stream mode.
- Most recent enhancement to SMF is Streaming support:
 - Provides real time access to SMF records in a wraparound buffer.
 - Unlike existing IEFU8x exits, no APF authorization is required AND there is no ability to modify or discard records.
 - Access is controlled using SAF (not possible with the IEFU8x exits).
 - Provides an official API to connect, retrieve, and disconnect.
 - Setup effort is trivial.
 - Interesting model if you have some SMF records with a VERY short shelf life – potentially no need to harden them to data sets.

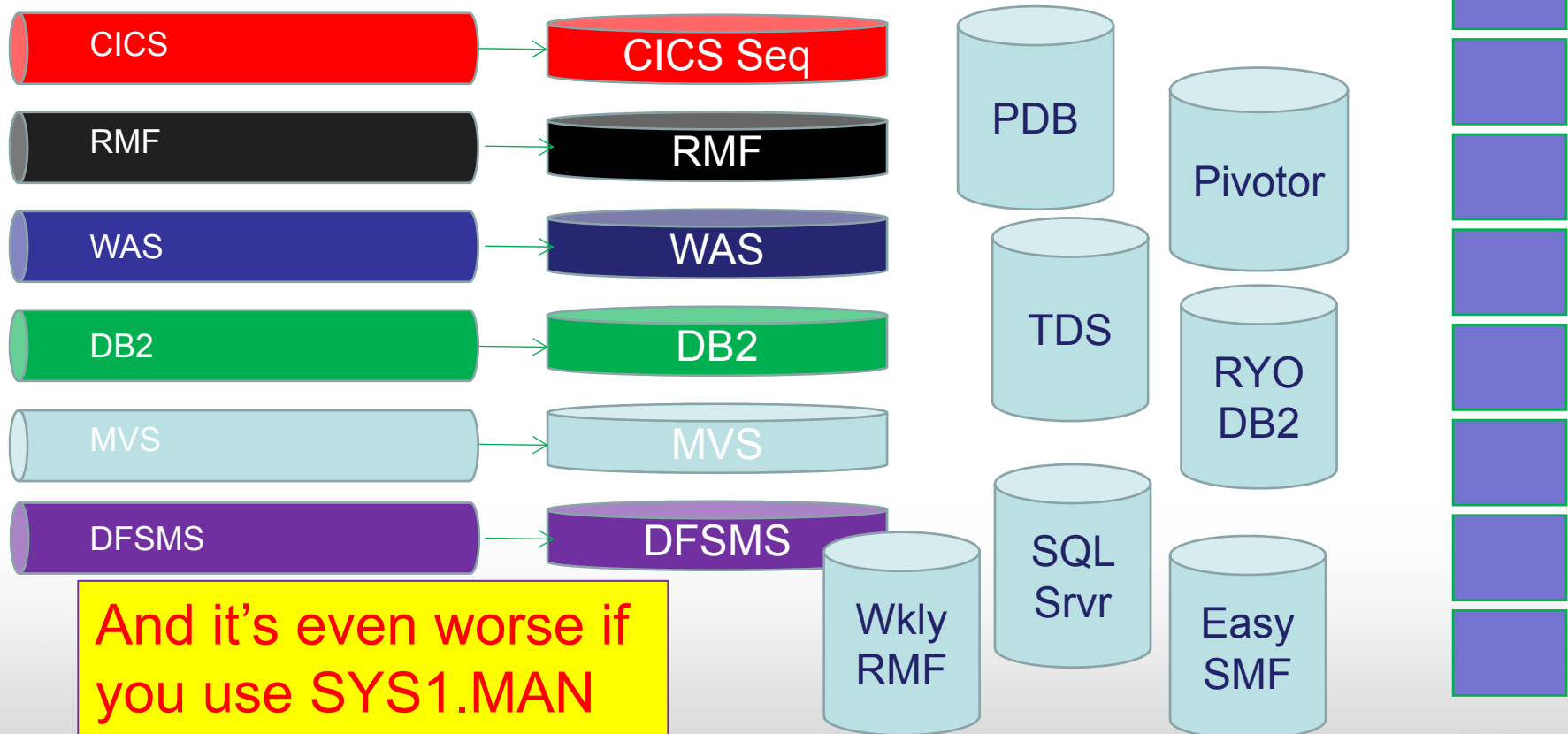
Be careful how you define logstreams

SMF Lifecycle



Get APAR OA49157 if you read log streams with SUBSYS=LOGR



- Regardless of where you write SMF data to, most sites then further post-process it



And it's even worse if you use SYS1.MAN

SMF and the Business World



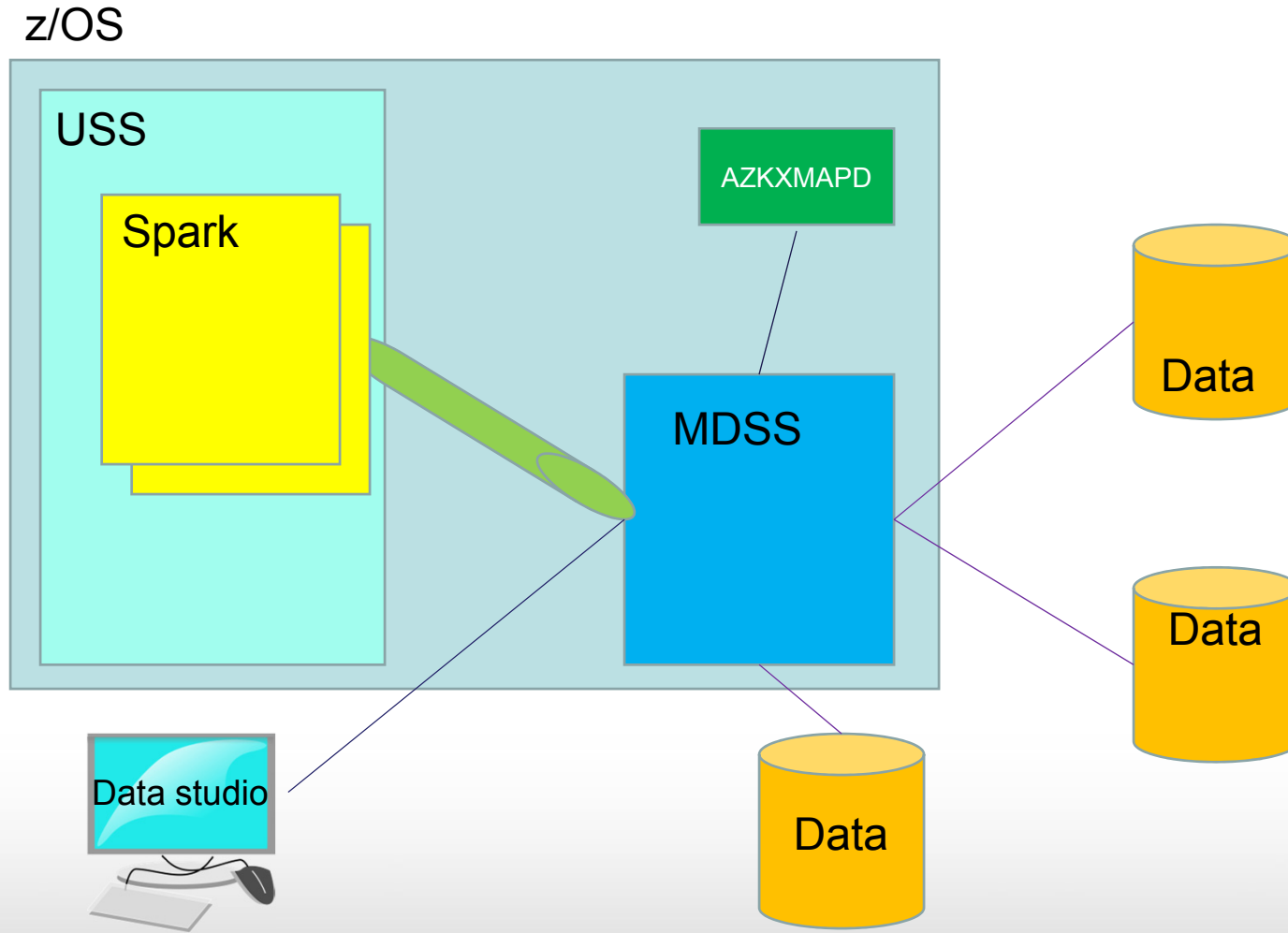
- In parallel with all of these happenings in the SMF world, your business users are demanding:
 - Lower costs. 
 - Higher application availability. 
 - *Preventing* unauthorized access (rather than being able to detect it after the fact).
- The IT department (YOU!) needs to address all these demands at the same time that the pool of experienced staff is shrinking.
 - And the skill set of the new generation of technicians is different (Java skills are now the norm).
- What is the common thread running through all this?? Exploiting analytics to mine your SMF data...

z/OS Platform for Apache Spark



- Enter.... The IBM z/OS Platform for Apache Spark ([5655-AAB](#)).
- Spark is an open source analytics platform. It is currently the most active open source project. The last Spark user conference had nearly twice as many attendees as SHARE *and* was sold out...
 - As an open source project, anyone can download Spark at no charge.
- The *recommended* way to get Spark for z/OS is via the IBM z/OS Platform for Apache Spark product. This delivers two components:
 - Apache Spark (version 1.5.2 at the time of writing).
 - Rocket Software's Mainframe Data Service for Apache Spark (MDSS).
 - MDSS includes Rocket Software's Data Service Studio
 - The option to get Service and Support.
- To avoid confusion, when I'm referring to 'IBM z/OS Platform for Apache Spark', I'll call it 'the Spark package' and when I'm referring to the Apache Spark component of that, I will just call it 'Spark'.

Spark Components



What is Spark?



- The Spark half of the Spark product could be considered as sort of analogous to CICS:
 - It provides an environment in which you can run ‘transactions’ (Java, Scala, or Python programs) THAT YOU (or a vendor) PROVIDE.
 - Spark is NOT a competitor to any existing product that consumes SMF data. It IS, potentially, a way for some of those products to access data quicker, at a lower cost, and provide more function.
 - It provides reporting and management capabilities.
 - It provides caches (Resilient Distributed Datasets (RDDs)) to hold data that is being analyzed by your programs, resulting in sub-second response times against what started as huge (multi-GB) sets of data.
- Where does all that data come from, and what does it have to do with SMF??

What is MDSS?



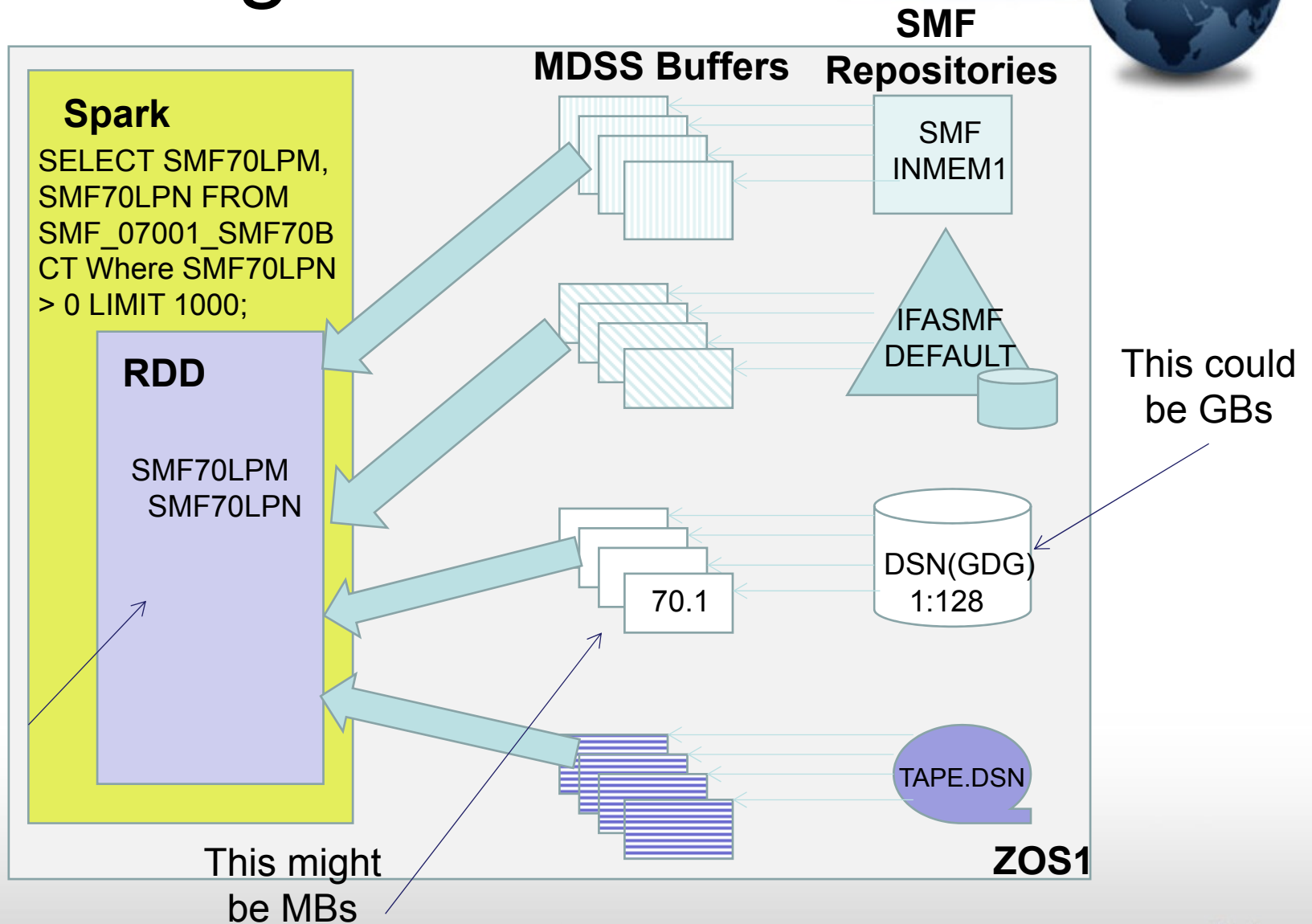
- That brings us to the other half of the z/OS Platform for Apache Spark product - MDSS:
 - MDSS is a subset of Rocket Software's Data Virtualization Services (DVS) product.
 - MDSS lets you map nearly any data set type (sequential, partitioned, VSAM, IMS, DB2, Adabas, SMF log streams, and SMF streaming) to SQL virtual tables.
 - You send SQL queries to MDSS from Spark and it retrieves the data using parallel I/Os (a LOT of work has been done to maximize parallelism wherever possible).
 - MDSS then feeds data back to Spark via JDBC (either locally or remote).
 - Spark can also communicate directly to DB2 or IMS using JDBC.

What Makes Spark so Special?



- Why are we (W&W) so excited about the Spark product?
 - It provides Spark *and* MDSS for **FREE!!** You only pay for S&S.
 - **ALL** processing for Spark itself, queries and programs that run under Spark, *and* MDSS is zIIP-eligible, so *no impact on your R4HA*.
 - MDSS *comes complete* with mappings (using standard SMF field names) for nearly all the most popular SMF records, *and* the list is constantly growing.
 - IBM are offering significant discounts for memory that is purchased specifically for use by Spark.
 - IBM is offering discounts for zIIPs purchased for use by Spark.
- This results in about the lowest cost work you will ever run on z/OS. AND, it gives me the same power and flexibility that I would have if my data was sitting in DB2, but without the disk space and CPU cost of loading it into DB2. It lets me (yes, even ME!) access SMF data sitting on a 10 year-old tape, AND SMF records that were just created 1 millisecond ago, using SQL SELECT statements!!
- Let's have a look at how all this works....

Spark Big Picture



Exploiting SMF Data with Spark



- IBM did NOT create the Spark product to allow us to play with SMF data.
- However, even though IBM's target market for Spark is business analytics, it doesn't ONLY support analytics. AND, SMF is an initial use of Spark that IBM is promoting.
- Java, Scala, or Python programs under Spark could potentially be used to replace traditional programs that run against SMF data in sequential data sets, log streams, or on tape (except that they would run on zIIPs rather than on general purpose CPs).
- Coding analytics programs is not for the faint-hearted, but *that can follow later*, when you get more familiar with the use of Spark.
- When you combine the new Streaming SMF capability with the ability to run complex analysis at a very low cost, we expect to see new uses of SMF data that we have not even imagined yet.

Getting Started



Because it is a BIG jump from SYS1.MAN data sets, to performing real time analytics against SMF data, we (W&W) are encouraging customers to implement this in a phased manner:

1. Implement MDSS to test your SQL and determine the value of SQL access to SMF (or other) data. It is also very useful for ad-hoc queries against SMF records.
2. Implement Spark and develop Java equivalents of existing home-grown programs that process SMF data.
3. Start doing real analysis against your 'historical' SMF data.
4. Start using real time SMF data to detect and address issues in real time. There are already products that are starting to do this, so this is likely to be used for site-specific applications, while products such as IBM Operations Analytics for z (IOAZ) provide generic solutions.

Remember, this is all free (until you decide if you want to pay for S&S), so you can back out at any time.

AND the MDSS ability to map data sets is not limited to SMF – you can map anything you like. If it can map complex data structures like SMF records, the majority of normal PS or VSAM files should be a piece of cake

What Would I Do With It?



- The ability to use SQL to access SMF data is interesting, but I already have products and RYO programs that process SMF data, so why would I be interested in Spark?
- 1) It all runs on zIIP – save all that GCP load
- 2) If I can run my queries against the original SMF repository (log streams or sequential data sets), I can save the disk space and CPU time I burn to move the data elsewhere today.
- 3) All my new people know Java, but not Assembler, maybe this is an opportunity to replace all my old homegrown programs that process my SMF data AND save some costs in the process.
- 4) Wouldn't it be nice if your products that format SMF records all ran against the same files?
- Remember, this is nearly free, so you don't NEED a long long list of exploiters and benefits to justify it.
- Let's look at an example of how you might use MDSS to replace an existing tool that processes SMF records.....

Real World Use of Spark



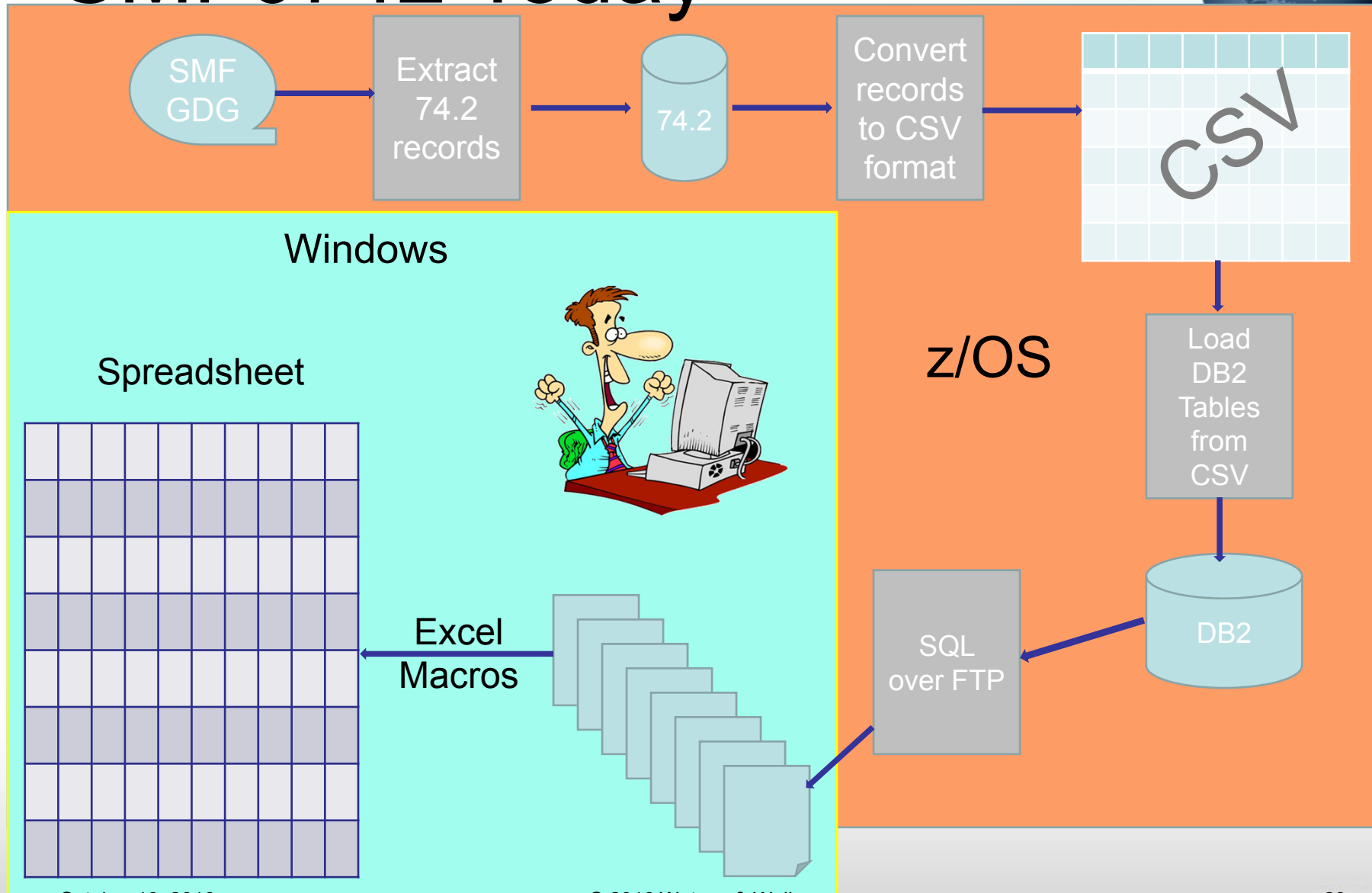
Mario Bezzi (IBM Italy) has a set of tools that he makes freely available that process SMF records and provide perspectives that are not available elsewhere. The resulting spreadsheet also provides 'exception' worksheets to help you easily detect out of line situations. There are many customers using this, and getting real value from it, so this is a 'real world' example.

Sheet Name	Description
WQIXC10M	SMF Data Report - Available System Data - System Overview
WQIXC11M	SMF Data Report - Available Path Data - System Overview
WQIXC12M	SMF Data Report - Available Member Data - System Overview
WQIXC20Q	System Activity - OutBound Traffic - Sysplex Overview
WQIXC21Q	System Activity - OutBound Traffic - System Overview
WQIXC22Q	System Activity - InBound Traffic - System Overview
WQIXC23Q	System Activity - Local Traffic - System Overview
WQIXC24Q	System Activity - OutBound Transport Class Overview
WQIXC25Q	System Activity - Local Transport Class Overview
WQIXC26D	System Activity - OutBound Transport Class Details
WQIXC27D	System Activity - Local Transport Class Details
WQIXC28D	System Activity - InBound System Details
WQIXC30E	System Exceptions - Abnormal System Status
WQIXC31E	System Exceptions - No OutBound Paths
WQIXC32E	System Exceptions - No OutBound Buffers
WQIXC33E	System Exceptions - No InBound Buffers
WQIXC34E	System Exceptions - No Local Buffers
WQIXC40D	Path Activity - Outbound Path Details
WQIXC41D	Path Activity - Inbound Path Details
WQIXC42Q	Path Activity - Inbound Path Structure Performances
WQIXC43Q	Path Activity - Inbound Path CTC Performances
WQIXC51E	Path Exceptions - OutBound Path
WQIXC52E	Path Exceptions - InBound Path
WQIXC53E	Path Exceptions - Abnormal Path
WQIXC60Q	Member Activity - Group Summary
WQIXC61Q	Member Activity - Group Overview
WQIXC62D	Member Activity - Group Details
WQIXC63Q	Member Activity - System Overview
WQIXC64D	Member Activity - System Details
WQIXC65D	Member Activity - Member Details

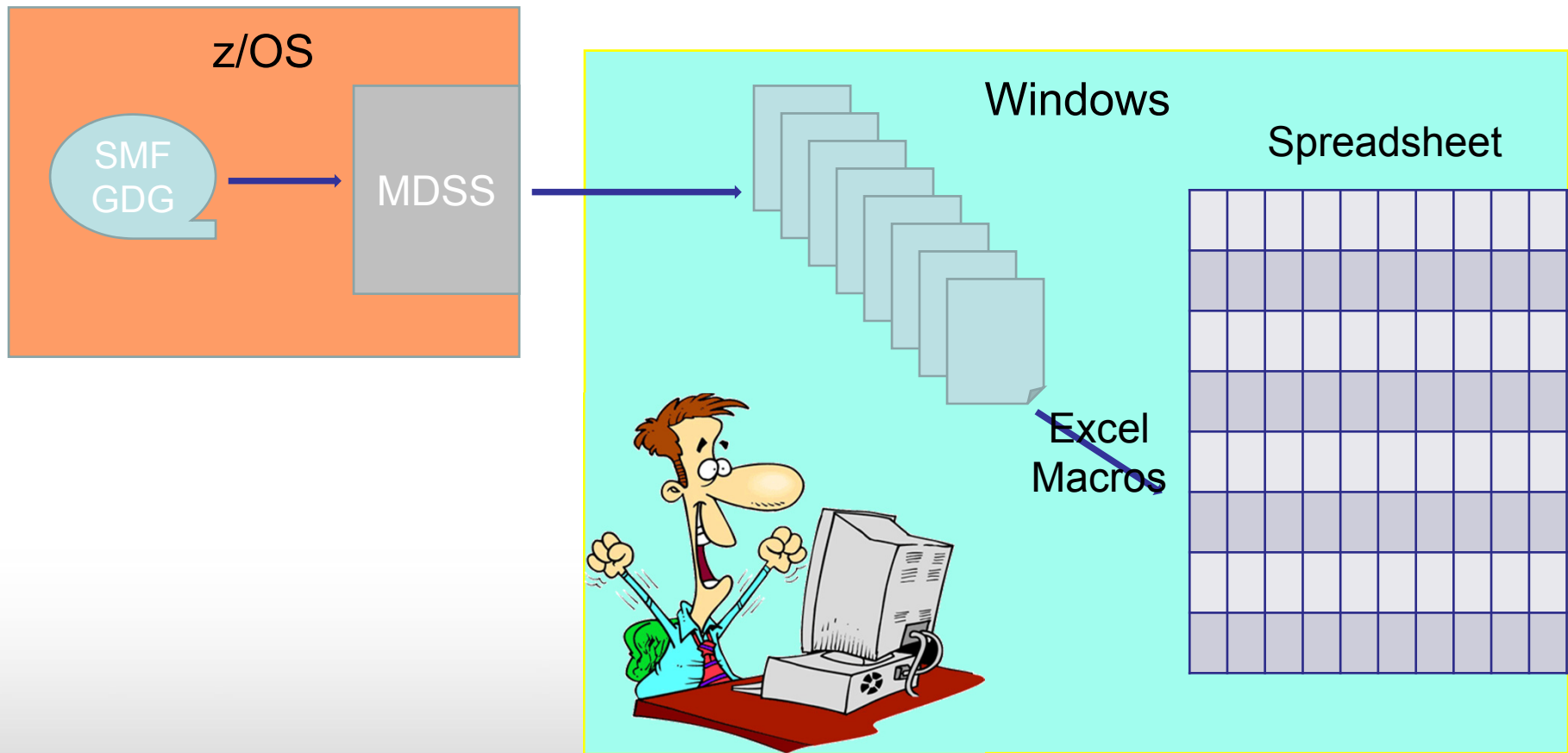
Each report is generated by an SQL query

This is just one example – he has a number of tools all based on the same model

SMF0742 Today



SMF0742 with Spark



SMF0742 with Spark



- All the data extraction and loading into DB2 currently runs on z/OS, using GCPs.
- Mario's SMF0742 package is supplied complete with the SQL queries.
- Queries are run over FTP and populate text files on your PC.
- Then run a spreadsheet macro to load the text files into Excel.
- Our objective is to replace all the z/OS end of processing with a query to MDSS based on slightly re-worked versions of Mario's code.
 - No Java code or any programming is required for this – the process to create the text files uses just SQL queries.
 - The processing in MDSS runs all on a zIIP, so no GCP MIPS consumed.

SMF0742 with Spark



- No Spark implementation is required at this point – it is all self-contained in MDSS.
- You have two options for how you would implement this:
 - Using the Data Studio. This provides more hand-holding. And you need to set up Data Studio anyway.
 - Using a batch interface (AZKXMAPD).
- For our example we will use Data Studio. (Installing Data Studio takes 5 minutes).
- Once you get familiar with MDSS and have some SQL queries to use as the base of your processing, you would move on and implement them in Spark.

Using MDSS & Data Studio



Click on Set Server to get started

Host: 192.168.1.61 (Port 1200)
Server: AZKS
Version: 01.01.00.0000

Set Server ...

Scan For More Servers...

- 192.168.1.61
 - Port 1200 (AZKS)
- rs01.rocketsoftware.com

Host: 192.168.1.61 Port: 1200
Userid: ibmuser User Password: *****

OK Cancel

Fill in the blanks

You are now connected to MDSS

Using MDSS & Data Studio



The screenshot displays the IBM Data Studio interface. The main window shows a tree view of 'Virtual Tables' for a server named 'AZKS'. The tables listed include:

- CLIENT_RETN
- DCL_CLIENT_INFO
- OPERLOG_MDB
- OPERLOG_MDB_MDB_CONTROL_OBJECT
- OPERLOG_MDB_MDB_TEXT_OBJECT
- OPERLOG_SYSLOG
- SMF_00000
- SMF_00600
- SMF_01400
- SMF_01400_IFGSMFUCB
- SMF_01500
- SMF_01500_IFGSMFUCB
- SMF_02600
- SMF_02600_SMF26AC1
- SMF_03000
- SMF_03000_SMF30ACS
- SMF_03000_SMF30AR
- SMF_03000_SMF30CAS
- SMF_03000_SMF30CDS
- SMF_03000_SMF30CMP
- SMF_03000_SMF30DR

The tables SMF_03000_SMF30ACS, SMF_03000_SMF30AR, SMF_03000_SMF30CAS, SMF_03000_SMF30CDS, SMF_03000_SMF30CMP, and SMF_03000_SMF30DR are highlighted with a red box. A blue callout box with white text points to this list, stating: 'Click on SQL->Data->AZKS->Virtual Tables to get a list of all the tables that map the SMF records. This shows ALL the virtual tables and subtables (subtables contain repeating sections in SMF records).' The SQL editor on the right contains a sample SQL statement: 'select SMF from SMF'. The bottom panel shows the 'SQL Results' tab with a list of DVS_SERVER calls.

Using MDSS & Data Studio



The screenshot displays the IBM Data Studio interface. On the left, the 'Data' pane shows a tree view of the database structure. A red box highlights the table 'SMF_07001' and its columns: SMF_LEN [INTEGER], SMF_ZERO [INTEGER], SMF_FLAG [VARCHAR(8)], SMF_RTY [SMALLINT], SMF_TIME [TIMESTAMP], SMF_SID [VARCHAR(4)], SMF_SSI [VARCHAR(4)], SMF_STY [SMALLINT], SMF_SEQN [VARCHAR(10)], SMF70TRN [INTEGER], SMF70PRS [INTEGER], SMF70PRL [INTEGER], SMF70PRN [INTEGER], SMF70CCS [INTEGER], SMF70CCL [INTEGER], SMF70CCN [INTEGER], SMF70CPS [INTEGER], SMF70CPL [INTEGER], SMF70CPN [INTEGER], SMF70ASS [INTEGER], SMF70ASL [INTEGER], SMF70ASN [INTEGER], SMF70BCS [INTEGER], SMF70BCL [INTEGER], SMF70BCN [INTEGER], SMF70BVS [INTEGER], and SMF70BVL [INTEGER].

The main editor window shows a SQL script with a blue callout box containing the text: "Click on table name to get a list of all the columns in that table. Column names generally match the field name in the SMF manual." Below the script, the 'SQL Results' pane displays a table with 17 rows and 13 columns. The columns are: SMF_LEN, SMF_ZERO, SMF_FLAG, SMF_RTY, SMF_TIME, SMF_SID, SMF_SSI, SMF_STY, SMF_SEQN, SMF70TRN, SMF70PRS, SMF70PRL, and SMF70PRN. The first row shows values: 0, 4860, 2, 11011111, 70, 2009..., PRD1, RMF, 1, ..., 7, 164, 184.

	SMF_LEN	SMF_ZERO	SMF_FLAG	SMF_RTY	SMF_TIME	SMF_SID	SMF_SSI	SMF_STY	SMF_SEQN	SMF70TRN	SMF70PRS	SMF70PRL
0	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
1	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
2	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
3	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
4	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
5	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
6	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
7	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
8	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
9	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
10	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
11	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
12	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
13	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
14	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
15	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
16	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184
17	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184

Columns Group: 1 of 3 Columns per group: 25
384 rows SQL Messages

Using MDSS & Data Studio



The screenshot shows the Data Studio interface with a SQL query editor at the top right and a results table at the bottom right. A green callout box points to an export arrow icon in the results table area.

SQL Query:

```
-- This is the quintessential sample SQL statement that is
-- used with DVS to demonstrate
-- tradition, we have preserved
-- Studio as well.

-- To run this, do the following:
-- 1) From your server view,
--    has the sample tables
--    choose "Create DSN From
--    a new DSN for you.
-- 2) Highlight the sql statement
-- 3) Press F5 to execute the sql.

-- Remember that you can always press F1 to get help.

select *
from SMF_07001
```

Results Table:

	SMF_LEN	SMF_ZERO	SMF_FLAG	SMF_RTY	SMF_TIME	SMF_SID	SMF_SSI	SMF_STY	SMF_SEQN	SMF70TRN	SMF70PRS	SMF70PRL	S
0	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
1	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
2	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
3	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
4	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
5	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
6	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
7	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
8	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
9	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
10	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
11	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
12	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
13	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
14	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
15	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
16	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1
17	4860	2	11011111	70	2009...	PRD1	RMF	1	...	7	164	184	1

Columns Group: 1 of 3 Columns per group: 25
384 rows SQL Messages

Now enter your SQL statement in the top right window. When you are ready to test it, press F5.

Results are presented in bottom right part of desktop – are they what you expect?

If you want to export to a CSV, click on the little export arrow

Using MDSS & Data Studio



- That's all there is to using Data Studio. All you need to know is a little (VERY little) SQL.
- Now let's look at what was involved in converting Mario's tool to run in MDSS instead.

Before

Count the number of CF-only log streams

```
select
  smf88lfl,
  (count(distinct(smf88lsn))) as CFOnly_Isn_c
from SMFDATA.SMF0881_LGRDATA
where int(left(smf88lfl,1)) = 0
group by smf88lfl;
```

After

Count the number of CF-only log streams

```
select
  smf88lfl,
  (count(distinct(smf88lsn))) as CFOnly_Isn_c
from SMF_08801 a0
join SMF_08801_SMF88LSD a1
  on a0.CHILD_KEY = a1.PARENT_KEY
where int(left(smf88lfl,1)) = 0
group by smf88lfl;
```

Using MDSS & Data Studio



- How about a slightly more complex query?

Before

Count the number of structure rebuild events

select

```
smf88syn, smf88lsn, smf88ltd, smf88lfl, smf88grp, smf88stn,  
smf88eri, smf88lwb,  
smf88swb, smf88ldb, smf88lwi, smf88sc1, smf88sc2, smf88sc3,  
smf88eo, smf88eds, smf88efs, smf88esf, smf88eri, smf88ett,  
smf88etf,  
smf88sib, smf88sii, smf88sab, smf88sai
```

from SMFDATA.SMF0881_LGRDATA

where smf88eri > 0

order by smf88eri desc;

After

Count the number of structure rebuild events

select

```
smf88syn, smf88lsn, smf88ltd, smf88lfl, smf88grp, smf88stn,  
smf88eri, smf88lwb,  
smf88swb, smf88ldb, smf88lwi, smf88sc1, smf88sc2, smf88sc3,  
smf88eo, smf88eds, smf88efs, smf88esf, smf88eri, smf88ett,  
smf88etf,  
smf88sib, smf88sii, smf88sab, smf88sai
```

from SMF_08801 a0

join SMF_08801_SMF88LSD a1

on a0.CHILD_KEY = a1.PARENT_KEY

join SMF_08801_SMF88ESD a2

on a0.CHILD_KEY = a2.PARENT_KEY

join SMF_08801_SMF88SSD a3

on a0.CHILD_KEY = a3.PARENT_KEY

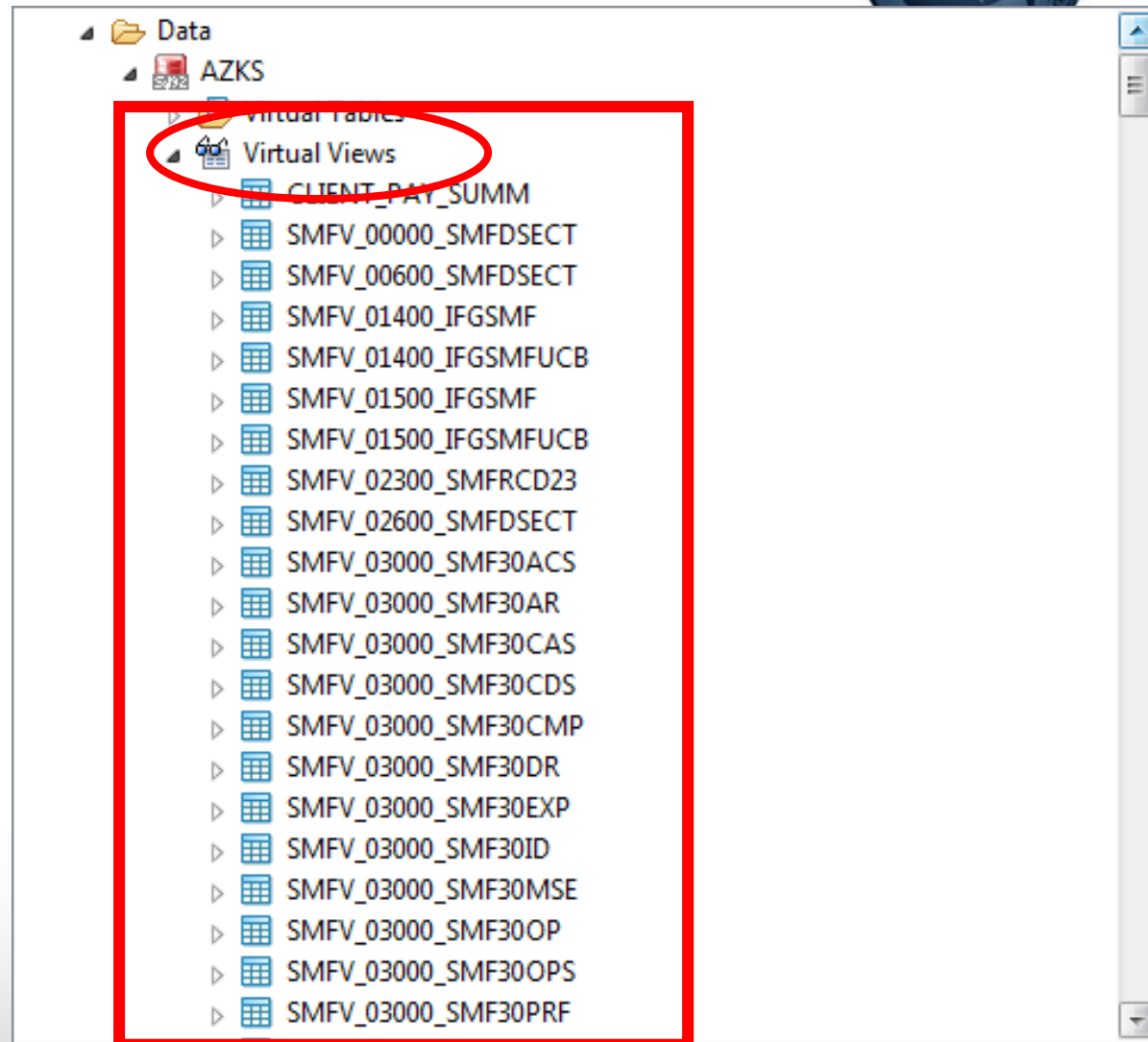
where smf88eri > 0

order by smf88eri desc;

Using MDSS & Data Studio



To save you from having to deal with all those pesky JOINS, MDSS also provides Views, which implement the JOINS for you.



Using MDSS & Data Studio

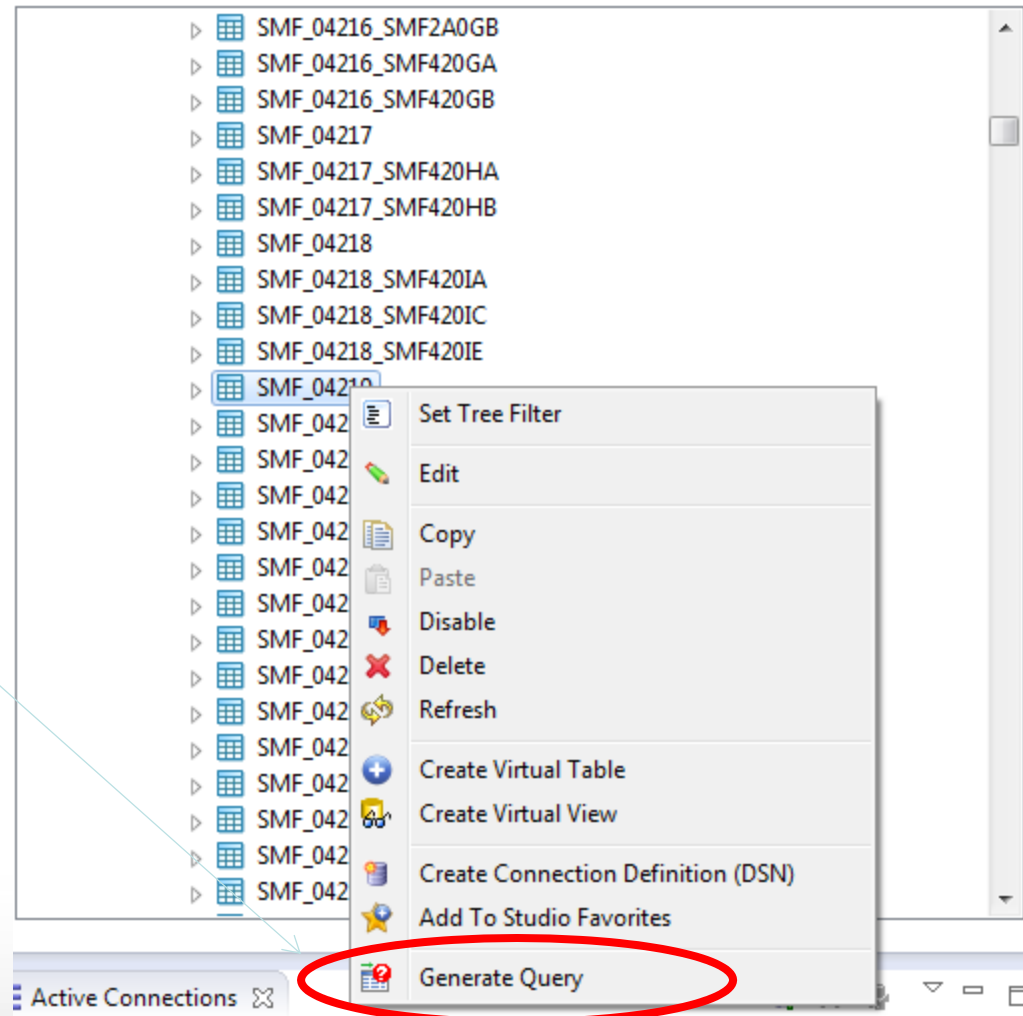


- To 'automate' the processing of several queries, you can have multiple Select statements in the source pane, swipe them all, and press F5.
- All queries will be run and the output placed back in multiple tabs (1 per Select) in the SQL Results area of the screen.
- However, using Data Studio, saving each tab to a CSV is a manual process.
- If you wanted, you could test your SQL statements in Data Studio, then copy/paste them into a batch job that runs AZKXMAPD, and write the output from each select to a separate member of a PDS, then just FTP them down to your PC and process them with the spreadsheet macro.
- All of this with no Java, and no need for Spark.
 - For more complex reports, we will wait until Phase 2, where we set up Spark and wrap some Java around the SQL statements that you have tested using MDSS.

Using MDSS & Data Studio



For those as SQL-illiterate as me, you can right-mouse on a table name, and click on Generate Query at the bottom of the list



Using MDSS



This generates a Select for all the columns in that table – just delete out the ones you don't want. Then add a WHERE to file, and an ORDER BY and you are done.

```
-- Name      : SMF_07001
-- Catalog   : null
-- Schema    : AZKSQL
-- Remarks   : DATA - SMFDATA
-- Tree Location: 192.168.1.61/1200/SQL/Data/AZKS/Virtual Tables/SMF_07001
-- The sql statement:
SELECT SMF_LEN, SMF_ZERO, SMF_FLAG, SMF_RTY, SMF_TIME, SMF_SID, SMF70XNM, SMF70SNM, CHILD_KEY, BASE_KEY
FROM SMF_07001 LIMIT 1000;
```

```
-----
-- This statement will return all rows and all columns from the
-- following table:
-- Name      : SMF_04219
-- Catalog   : null
-- Schema    : AZKSQL
-- Remarks   : DATA - SMFDATA
-- Tree Location: 192.168.1.61/1200/SQL/Data/AZKS/Virtual Tables/SMF_04219
-- The sql statement:
SELECT SMF_LEN, SMF_ZERO, SMF_FLAG, SMF_RTY, SMF_TIME, SMF_SID, SMF_SSI,
       SMF_STY, SMF_SEQN, SMF42NT, SMF42OPS, SMF42LPS, SMF42NPS, SMF42JN1, SMF42JN2,
       SMF42JN3, SMF42JN4, SMF42JN5, SMF42JN6, SMF2AJN1, SMF2AJN2, SMF2AJN3, SMF2AJN4,
       SMF2AJN5, SMF2AJN6, CHILD_KEY, BASE_KEY
FROM SMF_04219 LIMIT 1000;
```

Using MDSS

The screenshot shows the 'Data Service' preferences dialog box in IBM Data Service Studio. The 'SQL Results Max Rows' and 'SQL Results Max Bytes' fields are highlighted with a red box. A red circle highlights the 'Properties' button in the bottom right corner of the dialog. Two callout boxes provide instructions: one on the left says 'If you get a message saying "Max Rows Reached" and you want more data, click here and select properties', and one on the right says 'Then change these values'. The 'SQL Results Max Rows' is set to 1000 and 'SQL Results Max Bytes' is set to 10000000. The 'SQL Generate Query Behavior' is set to 'Generate query and issue user prompt'. The 'Hex Encoding' is set to 'UTF-8'. The 'Studio Fixed Width Font' is set to 'Courier New-regular-9'. The 'New Connection (DSN) Naming Pattern' is set to '{Subsystem}'. The 'Studio HTTP Debug Option' is set to 'Normal'. The 'Enable Server Tracing of Studio Calls' checkbox is checked. The 'Properties' button is circled in red.

Data Service

General settings for Data Service Studio.

Enable Server Tracing of Studio Calls

Studio HTTP Debug Option: Normal

Studio Fixed Width Font: Courier New-regular-9 [Font...] [Default]

Hex Encoding: UTF-8

New Connection (DSN) Naming Pattern: {Subsystem}

SQL Generate Query Behavior: Generate query and issue user prompt

SQL Results Max Rows: 1000

SQL Results Max Bytes: 10000000

Properties

Virtual Table:
TIME, SMF_SID,

Columns from the
Virtual Table
TIME, SMF_SID,
SMF42NPS, SMF
, SMF2AJN2, S

Next steps



- Note that MDSS is a *test* platform – it is not intended for production levels of data and does not provide all the performance-enhancing features that are provided by Spark.
 - But, it is ideal for adhoc queries when you just need to pull out some fields from particular SMF records because of the range of record types it supports
- Having tested that your SQL behaves as you expect, you can then copy/paste it to a Java or Scala program to run it under Spark.

Next steps



- Spark is really designed for highly parallel, data-intensive, analytics work.
- BUT, until you are ready to expand to that, it is a very effective platform to run your traditional SMF reports.
 - Running cost is as close to zero as you will get.
 - No need to move your SMF data to other platforms.
 - Keeping all SMF data on z/OS is ideal for when you want to expand to include real time analytics using SMF Streaming in the future.

'Concerns'/Comments



- This is Version 1, Release 1, so expect a few teething problems.
- Documentation is more reference manual than User's Guide.
- Some critical information was not described (for example, how to set up the table to DSN mappings) in any of the manuals.
- Some of the code is a little 'rough' yet – e.g. if you try to go into the MDSS ISPF application before the AZKS STC finishes initializing, you get an ABEND0C3.
- Because MDSS runs multiple reads in parallel, you need to think about using the ORDER BY to get the rows into the desired sequence.
- If you do NOT pay for S&S, you CAN retrieve PTFs for Spark/MDSS, but you do NOT get access to Rocket Software's Knowledgebase. You might want to sign up for S&S at least until you get up and running.
- Despite the above issues, the support that we received was fantastic. I managed to get it working and I know NOTHING about SQL...
- Spark is not from the mainframe world, so it doesn't have a lot of the controls and gentlemanly behavior that we have come to expect (remember, where Spark comes from, it is the only application running on that hardware). Also, currently doesn't appear to be any z/OS-specific documentation for setting up the Spark part of the package.

The Future



- IBM are putting a lot of investment into Spark on z/OS, so expect to see enhancements based on traditional z/OS capabilities (WLM, for example) that address some of Spark's shortcomings.
 - Rocket are adding new functions to MDSS nearly every week.
- The z/OS Platform for Spark Apache is currently based on Spark 1.5.2. The latest available Spark is 2.0, so expect to see the z/OS version move to 2.0 in the reasonably near future.
- With such a compelling cost case, speak to the vendors of your SMF-consuming products to encourage them to add support for Spark.
- Rocket are actively seeking partners to create samples of SMF programs for Spark. These will be made available to all users, with users encouraged to share their programs.

References



- IBM Redbook [SG24-8325](#), *Apache Spark implementation on IBM z/OS*
- Spark bookshelf: <http://www-03.ibm.com/systems/z/os/zos/library/bkserv/v2r2pdf/#AZK>
- Installation White Paper [WP102609](#), *Installing IBM z/OS Platform for Apache Spark*.
- Cheryl Watson's Tuning Letter 2016 No. 1 article about new SMF Streaming capability.
- Cheryl Watson's Tuning Letter 2016 No. 3 about Spark and SMF.
- www.watsonwalker.com will be adding a repository that will include sample SQL queries and Java programs for use with Spark and SMF.
- For fixes, search on component ID 5655AAB02.

Prerequisites



- Spark runs on zEC12 and later. But the significantly lower memory costs on z13, as well as features such as SMT, SIMD, and SMC-D, make it a more logical platform to deploy Spark.
- Software requirements:
 - z/OS 2.1
 - IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8 Refresh 2 Fix Pack 10.
 - Bourne Again Shell (bash) version 4.2.53, or later.
- There is no initial or MLC charge for the z/OS Platform for Apache Spark. The S&S cost is a flat price, per CPC, and is the same regardless of the size of the CPC or how many LPARs you run it in.

Summary



- Given the business requirements for increased security, reduced software costs, and the imperative to get additional value from all the data in your shop, the z/OS Platform for Apache Spark seems to have arrived at exactly the right time.
- Implementing the MDSS half of the product is fast and easy and gives a flavor of what can be achieved when you combine SMF, SQL, and Java. The real value comes from exploiting the capabilities of Spark, but that implementation is more complex.
- It will probably be a while before we see vendor applications that exploit Spark support of SMF, but in the interim, we recommend installing Spark *now* and start building your experience with it in an incremental manner.
 - Also take the opportunity to see if Spark can help with applications other than SMF.
- Rumors are that IBM has a long list of customers queueing up to run Proof Of Concepts. I know of at least one customer that is already running Spark in production.

Questions?



- If you have any questions, or would like the latest version of the slides, please send me an email at frank@watsonwalker.com

Thanks!!!

- Please complete an evaluation form – Session 525



Supplemental Slides

Implementing MDSS

Implementation



- Even though they are delivered as a single product, Spark and MDSS are effectively separate products, delivered with a lot of integration to glue them together.
- Installing MDSS and getting it up and running is simple. Implementing the Spark half of the equation is more complex and probably requires more add-ons to deliver a complete environment.
 - You can run Spark queries from the Bash shell in TSO.
 - But it is probable that ‘developers’ (more likely data scientists) will require add-on functions like Jupyter Notebooks to provide a full development environment.
- However, you can implement this in a phased manner:
 1. Set up MDSS and test your SQL statements using Data Studio.
 2. Implement Spark and use it to run Java programs against your SMF data, replacing existing home-grown SMF processing programs.
 3. Implement SMF Streaming and integrate that with analytics programs and automation products to identify pending problems or exposures and take proactive action.

Implementing MDSS and Data Studio



- MDSS is installed into standard PDSs and PDSEs (AZK.SAZK*)
- APF authorize the MDSS load libraries.
- Customize a REXX to invoke the MDSS ISPF application.
- Download the Data Studio from the SAZKBIN data set.
- Copy the Started Task JCL (AZKS) into your Proclib and do a little customization.
- MDSS 'Parmlib' is actually implemented as an exec called AZKSIN00 in the SAZKEXEC data set.
- S AZKS. Make sure you get message: AZK3253I Data Service for Apache Spark version 01.01.00 build xxxx subsystem AZKS initialization complete.
- Invoke the AZKS REXX to get into the MDSS ISPF application.
- Set up some Global Variables to map virtual SMF tables to data sets.

Implementing MDSS



IBM z/OS Platform for Apache Spark - Primary Option Menu

Interface Facilities:

- 1 ACI
- 2 Adabas
- 3 DB2
- 4 IMS
- 5 VSAM/Sequential

SSID : AZKS
Version : 01.01.00
Date : 16/09/16
Time : 13:49

Server Administration:

- A Remote User - Manage Remote Users
- B Server Trace - Server Trace Facility
- C AZK Admin. - Manage Data Virtualization Server
- D Data Mapping - Data Mapping Facility
- E Rules Mgmt. - Event Facility Management
- F Monitor - Monitor Server Activity

IBM* Rocket**
Licensed Materials - Property of IBM. 5655-AAB
Copyright IBM Corporation 1991 - 2016 All Rights Reserved.
Copyright Rocket Software, Inc. 1991 - 2016 All Rights Reserved.
*Trademark of International Business Machines
**Trademark of Rocket Software, Inc.

Option ==> _____

Implementing MDSS



Server Management Menu

SSID: AZKS

Option ==>

- | | | |
|----|---------------|--|
| 1 | ISPF Session | - Display and modify ISPF/AZK session parameters |
| 2 | AZK ParmS | - Display and modify AZK main task parameters |
| 3 | AZK Blocks | - Display formatted AZK control blocks |
| 4 | AZK Stats | - Display AZK product statistics |
| 5 | AZK Tokens | - Display and control product tokens |
| 6 | AZK Modules | - Display product module information |
| 7 | AZK Tasks | - Display product tasks |
| 8 | AZK IP Tree | - Display the IP address tree |
| 9 | AZK Prcs Blks | - Display the Cross Memory Process Blocks |
| 11 | AZK Copies | - Display information about each copy of the product |
| 12 | AZK Storage | - Display virtual storage information |
| 13 | Trace Archive | - Server Trace Archive Facility |
| 14 | AZK Group | - Display all remote users in a group |
| 15 | NLS Tables | - Display National Language Support tables |
| 16 | Link | - Display Link Tables |
| 17 | RRS | - Display RRS Facilities |
| 18 | SOM | - Display and control Security Optimization and Management |
| 19 | JVM Admin | - Display and manage Java Virtual Machines |
| 20 | SSL Admin | - Display and manage AT-TLS configuration |
| 21 | SMF Real-Time | - Display and manage the SMF Real-Time tasks |

Implementing MDSS



----- Parameter Groups ----- Row 1 of 24
LCs: D Display F Format P Print CB S Show CB

Parameter Group	Group Description
PRODACI	ACI CONFIGURATION PARAMETERS
PRODADABAS	ADABAS PARAMETERS
PRODAPPCMVS	APPC/MVS PARAMETERS
PRODBROWSE	TRACE BROWSE PARAMETERS
PRODCOMM	COMMUNICATIONS PARAMETERS
PRODDMF	DMF (DATA MAPPING FACILITY) PARAMETERS
PRODEVENT	EXCEPTION EVENT PARAMETERS
PRODGLV	GLOBAL VARIABLE PARAMETERS
PRODIMS	IMS PARAMETERS
PRODLOGGING	LOGGING PARAMETERS
PRODMESSAGES	MESSAGES
PRODMODULES	MODULES
PRODPARM	GENERAL PARAMETERS
PRODREXX	REXX PARAMETERS
PRODRPC	RPC PARAMETERS
PRODRRS	RESOURCE RECOVERY SERVICES PARAMETERS
PRODSECURITY	SECURITY PARAMETERS
PRODSEF	SEF PARAMETERS
PRODSQL	SQL PARAMETERS
PRODSTOR	STORAGE PARAMETERS
PRODTRACE	TRACE PARAMETERS
PRODWLM	WLM SUPPORT PARAMETERS
PRODALL	ALL PRODUCT PARAMETERS
OBSOLETE	OBSOLETE PRODUCT PARAMETERS

All parameters are a member of a Parameter Group

End

Command ==> _

Scroll ==> PAGE

Implementing MDSS



```
----- Parameters ----- Scr 1 Row 1 of 92
LCs: D Display E Edit F Format P Print CB S Show CB

Parameter
Description
API REDIRECTOR SV
ATTENUATION FACT
AUTO-ADJUST TSO
AUTOMATIC CANCEL
AVAILABLE VS FOR
BYPASS SYSTEM NAM
CHECK THE STATUS OF EACH SESSION
d CLIENT CANCEL WAIT TIME VALUE
CLIENT TASK QUIESCE DELAY
CPU/WAIT LIMITS CHECKING INTERVAL
DEFAULT DBCS TABLE NAME
DIRECTED LOAD DDNAME
ENFORCE SEM CONCURRENT THREAD MA
ENFORCE TARGET ODBC/JDBC/J2CA THI
ENFORCE TARGET WEB SERVICE THREA
ERLY SUBSYSTEM
ERROR CPU TIME VALUE
ERROR WAIT TIME VALUE
FAIL CPU TIME VALUE
FAIL EXCLUSIVE LOCK TIME VALUE
FAIL SHARE LOCK TIME VALUE
FAIL SQL CPU TIME VALUE
FAIL UPDATE LOCK TIME VALUE
FAIL WAIT TIME VALUE
FORCE INITIALIZATION OF UNICODE CONV SVCS
GLOBAL REGISTRY COMPATIBILITY

Parameter
Value
NO
0.2
2147483647
YES
60 SECONDS
..
NO
500 MILLISECONDS
15 SECONDS
15 SECONDS
..
..
NO
NO
NO
NO
0 SECONDS
0 SECONDS
0 SECONDS
0 SECONDS
0 SECONDS
0 SECONDS
0 SECONDS
0 SECONDS
NO
NO

Command ==>
Scroll ==> PAGE
```

For an explanation of the meaning of a Parm, enter D beside Parm Description

Parms in red can be changed dynamically by simply overtyping the value

Implementing MDSS



Menu Utilities Compilers Help

```
BROWSE      Parameter Information                               Line 0000000000 Col 001 080
***** Top of Data *****
PARM        CANCELWAITTIME

MESSAGE     CLIENT CANCEL WAIT TIME VALUE

EXPLAIN     The CANCELWAITTIME parameter controls how long the
            product waits between client thread termination events
            during product shutdown or at any other time. This
            includes checking limits for each client thread. Note
            that the product automatically terminates client threads
            during product termination and if they have exceeded
            installation specified limits.
            Some IBM products cannot handle a large number of thread
            termination events in a short period of time. To prevent
            problems, the product throttles client thread terminations.
            The CANCELWAITTIME parameter is the delay between each
            client thread termination initiated by the product.
***** Bottom of Data *****
```

Command ==> _____ Scroll ==> PAGE

Implementing MDSS



- With over 1000 parameters in total, you have complete control over MDSS. However, with so many parameters spread over so many groups, it can sometimes be a challenge to find the right parameter to implement the change you want to make.
- To find the parm you're looking for, select PRODALL, scroll right (PF11), then do Sort Name – this will sort all the parms into name order. Then issue an 'L parmname' to find the parm.
- Hopefully IBM or Rocket will provide some guidance about which are the parms you really need to concentrate on.

Implementing MDSS



Server Management Menu

SSID: AZKS

Option ==> _____

- | | |
|------------------|--|
| 1 ISPF Session | - Display and modify ISPF/AZK session parameters |
| 2 AZK Parms | - Display and modify AZK parameters |
| 3 AZK Blocks | - Display format information |
| 4 AZK Stats | - Display AZK performance statistics |
| 5 AZK Tokens | - Display and control AZK tokens |
| 6 AZK Modules | - Display product modules |
| 7 AZK Tasks | - Display product tasks |
| 8 AZK IP Tree | - Display the IP tree |
| 9 AZK Prcs Blks | - Display the Cross memory Process blocks |
| 11 AZK Copies | - Display information about each copy of the product |
| 12 AZK Storage | - Display virtual storage information |
| 13 Trace Archive | - Server Trace Archive Facility |
| 14 AZK Group | - Display all remote users in a group |
| 15 NLS Tables | - Display National Language Support tables |
| 16 Link | - Display Link Tables |
| 17 RRS | - Display RRS Facilities |
| 18 SOM | - Display and control Security Optimization and Management |
| 19 JVM Admin | - Display and manage Java Virtual Machines |
| 20 SSL Admin | - Display and manage AT-TLS configuration |
| 21 SMF Real-Time | - Display and manage the SMF Real-Time tasks |

There is no message manual, but message information can be viewed in the ISPF application (in the Parms option)

Implementing MDSS



----- Parameters ----- Scr 1 Row 27 of 1719
LCs: D Display E Edit F Format P Print CB S Show CB

Parameter Description	Parameter Value
SEVERITY OF MESSAGE ID AZK0028	'S'
SEVERITY OF MESSAGE ID AZK0029	'S'
SEVERITY OF MESSAGE ID AZK0030	'S'
SEVERITY OF MESSAGE ID AZK0031	'E'
SEVERITY OF MESSAGE ID AZK0032	'S'
SEVERITY OF MESSAGE ID AZK0033	'S'
SEVERITY OF MESSAGE ID AZK0034	'S'
SEVERITY OF MESSAGE ID AZK0035	'S'
SEVERITY OF MESSAGE ID AZK0036	'S'
d SEVERITY OF MESSAGE ID AZK0037	'E'
SEVERITY OF MESSAGE ID AZK0038	'S'
SEVERITY OF MESSAGE ID AZK0039	'S'
SEVERITY OF MESSAGE ID AZK0040	'S'
SEVERITY OF MESSAGE ID AZK0041	'S'
SEVERITY OF MESSAGE ID AZK0042	'I'
SEVERITY OF MESSAGE ID AZK0043	'H'
SEVERITY OF MESSAGE ID AZK0044	'S'
SEVERITY OF MESSAGE ID AZK0045	'S'
SEVERITY OF MESSAGE ID AZK0046	'W'
SEVERITY OF MESSAGE ID AZK0047	'I'
SEVERITY OF MESSAGE ID AZK0048	'E'
SEVERITY OF MESSAGE ID AZK0049	'I'
SEVERITY OF MESSAGE ID AZK0050	'W'
SEVERITY OF MESSAGE ID AZK0053	'S'
SEVERITY OF MESSAGE ID AZK0054	'H'
SEVERITY OF MESSAGE ID AZK0055	'E'

Command ==>

Scroll ==> PAGE

Implementing MDSS



IBM z/OS Platform for Apache Spark - Primary Option Menu

Interface Facilities:

1	ACI		
2	Adabas	4	IMS
3	DB2	5	VSAM/Sequential

SSID : AZKS
Version : 01.01.00
Date : 16/09/16
Time : 13:49

Server Administration:

A	Remote User	-	Manage Remote Users
B	Server Trace	-	Server Trace Facility
C	AZK Admin.	-	Manage Data Virtualization Server
D	Data Mapping	-	Data Mapping Facility
E	Rules Mgmt.	-	Event Facility Management
F	Monitor	-	Monitor Server Activity

IBM* Rocket**
Licensed Materials - Property of IBM. 5655-AAB
Copyright IBM Corporation 1991 - 2016 All Rights Reserved.
Copyright Rocket Software, Inc. 1991 - 2016 All Rights Reserved.
*Trademark of International Business Machines
**Trademark of Rocket Software, Inc.

Option ==> _____

Implementing MDSS



----- Server Data Mapping Facility -----

SSID: AZKS

```
0  Map Defaults
1  Map Display
2  Map Copy
3  Map Refresh
5  VSAM File Control

I  Initialize Catalog
S  Source Library Management
```

This is where you come to get information about 'Maps'.
A Map is how the record layout of one or more data sets are described to MDSS in SQL terms.
From a programmer perspective, maps are equivalent to tables and views.

Option ==> _

Implementing MDSS



----- Data Mapping Block ----- Scr 1 Row 27 of 498

LCs: P Print Map S Show Map D Disable E Enable K Delete X Display

Structure Name	Type	Status	MR	Language	At	-Modification-		USERID	Note
					Date	Time			
NEONCONV		Enabled	Y	Catalog	N	**/**/**	00:00	AI38KB1	
OPERLOG_		Enabled	Y	Sequential	Y	**/**/**	00:00	AI38RPW	
OPERLOG_		Enabled	Y	Sequential	Y	**/**/**	00:00	AI38RPW	
PIOCOMM		Enabled	Y	ACI	Y	**/**/**	00:00	AI38FAB	
PRIMARYK		Enabled	Y	Catalog	N	**/**/**	00:00	AI38KB1	
PROCCOLS		Enabled	Y	Catalog	N	**/**/**	00:00	AI38KB1	
ROUTINES		Enabled	Y	Catalog	N	**/**/**	00:00	AI38KB1	
SCHEMAS		Enabled	Y	Catalog	N	**/**/**	00:00	AI38KB1	
SMF_FILE		Enabled	Y	Sequential	Y	**/**/**	00:00	AI38KB1	
SMF_0000		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0060		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0140		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0150		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0260		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0300		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	View	Y	**/**/**	00:00	TSOSB	
SMF_0420		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0421		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0421		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0421		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	
SMF_0421		Enabled	Y	Sequential	Y	**/**/**	00:00	TSOSB	

SMF Virtual tables

Note that this is only a partial list. Only base tables are shown here – for a complete list, use the Data Studio..

Command ==> _

Scroll ==> PAGE

Implementing MDSS



IBM z/OS Platform for Apache Spark - Primary Option Menu

Interface Facilities:

- 1 ACI
- 2 Adabas
- 3 DB2
- 4 IMS
- 5 VSAM/Sequential

SSID : AZKS
Version : 01.01.00
Date : 16/09/16
Time : 13:49

Server Administration:

- A Remote User - Manage Remote Users
- B Server Trace - Server Trace Facility
- C AZK Admin. - Manage Data Virtualization Server
- D Data Mapping - Data Mapping Facility
- E Rules Mgmt. - Event Facility Management ←
- F Monitor - Monitor Server Activity

IBM* Rocket**
Licensed Materials - Property of IBM. 5655-AAB
Copyright IBM Corporation 1991 - 2016 All Rights Reserved.
Copyright Rocket Software, Inc. 1991 - 2016 All Rights Reserved.
*Trademark of International Business Machines
**Trademark of Rocket Software, Inc.

Option ==> _____

Implementing MDSS



----- Event Facility (SEF) Control -----

SSID: AZKS

- 1 Global Variables - Display and Update Global Variables
- 2 SEF Rule Management - Control SEF Event Procedures & Libraries
Show Selection Panel at Entry ==> Y
- 3 Interactive Command - Issue SEF, AZK., or Product Rexx Commands

'Event Facility' Rules control many aspects of MDSS operation.

We are interested in the 'VTB' rules which are used to map virtual tables to real data sets

Option ==> _

Implementing MDSS



- Selecting option 1 gives you this:

```
----- Display Global Variables ----- Row 1 to 1 of 1
LCs: S Show Subnodes      M Modify Value      X Hex Browse
      D Remove Node and Subnodes      P Drop Node      B Browse

Global Prefix: GLOBAL2

S Subnode Name      Nodes      Subnode Value
-----
SMFTBL2              5 NO VALUE ASSIGNED AT THIS LEVEL
**End**
```

- Make sure that Global Prefix is set to GLOBAL2.
- Then place an S beside SMFTBL2.

Implementing MDSS



```
----- Display Global Variables ----- Row 1 to 5 of 5
LCs: S Show Subnodes      M Modify Value      X Hex Browse
      D Remove Node and Subnodes              P Drop Node      B Browse

Global Prefix: GLOBAL2.SMFTBL2                BEWARE: DSNs are case-sensitive

S Subnode Name      Nodes      Subnode Value
-----
DEFAULT              0 WWSMF.TEST1.SMF.T7049914
GDGBASE              0 WWSMF.TEST1.SMF.D091516.T70.SORTED
GDG0                 0 WWSMF.ROGER.SMF.D071116.SORTED
GDG1                 0 WWSMF.ROGER.SMF.D071116.SORTED
SMF_07001            0 WWSMF.TEST2.D29SEP16.SORTED
**End**
```

Existing table name to DSN mappings

Defining a new table name to DSN mapping

```
Command ==> S SMF 07002 WWSMF.TEST2.D29SEP16.SORTED      Scroll ==> PAGE
```

Implementing MDSS



```
----- Event Facility (SEF) Event Procedure List Row 1 to 11 of 11
LCs: S ISPF Edit E Enable D Disable A Set Auto-Enable
     Z Reset Auto-Enable B Set Auto/Enable C Disable/Reset Auto
```

PDS Members for: **AZK.SAZKXVTB**

We are using rule AZKSMFT2 to map SMF table names to data

S	Member	Status	E	TYP	VV.MM	Create	ID
	AZKGALIA	DISABLED	N	***			
	AZKMDLVS	DISABLED	N	***			
	AZKMDSUB	DISABLED	N	***			
	AZKMDTBL	DISABLED	N	***			
	AZKSMFT1	DISABLED	N	***			
	AZKSMFT2	ENABLED	N	VTB	01.06	16/09/11 16/09/11	
	AZKSMFT3	DISABLED	N	***	01.02	16/09/12 16/09/12	
	AZKSUBAL	DISABLED	N	***			
	AZKSYSLG	DISABLED	N	***			
	AZKSYSL2	DISABLED	N	***			
	AZKVSAMP	DISABLED	N	***			
	End						

From this panel, you can edit a rule, disable, and enable rules

To add a rule, simply add a member to the data set

These members....

Reside in this data set

Use B to have the SMF rule automatically enabled after each AZKS restart.

There is also an option to specify the DSN on the table name in your SQL statement

Command ==> _

Scroll ==> PAGE

Implementing MDSS



IBM z/OS Platform for Apache Spark - Primary Option Menu

Interface Facilities:

- 1 ACI
- 2 Adabas
- 3 DB2
- 4 IMS
- 5 VSAM/Sequential

SSID : AZKS
Version : 01.01.00
Date : 16/09/16
Time : 13:49

Server Administration:

- A Remote User - Manage Remote Users
- B Server Trace - Server Trace Facility ←
- C AZK Admin. - Manage Data Virtualization Server
- D Data Mapping - Data Mapping Facility
- E Rules Mgmt. - Event Facility Management
- F Monitor - Monitor Server Activity

IBM* Rocket**
Licensed Materials - Property of IBM. 5655-AAB
Copyright IBM Corporation 1991 - 2016 All Rights Reserved.
Copyright Rocket Software, Inc. 1991 - 2016 All Rights Reserved.
*Trademark of International Business Machines
**Trademark of Rocket Software, Inc.

Option ==> _____

Implementing MDSS



```

----- Server Trace ----- 08:42:46 16 SEP 16      Cols 001 060
HH:MM:SS HOST NAME -----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----
08:42:46 N/A      SETSOCKETOPTION EXECUTED - SOCK 0000 - SET SOCKET OPTION COM
08:42:46 N/A      SETSOCKETOPTION EXECUTED - SOCK 0000 - SET SOCKET OPTION COM
08:42:46 N/A      SETSOCKETOPTION EXECUTED - SOCK 0000 - SET SOCKET OPTION COM
08:42:46 N/A      SETSOCKETOPTION EXECUTED - SOCK 0000 - SET SOCKET OPTION COM
08:42:46 N/A      BIND EXECUTED
08:42:46 N/A      GETCLIENTID EXECUTED - GET CLIENT ID COMPLETED
08:42:46 N/A      GETSOCKNAME EXECUTED - GET SOCKET NAME COMPLETE
08:42:46 N/A      GETHOSTID EXECUTED - GET HOST ID COMPLETED
08:42:46 N/A      GETHOSTNAME EXECUTED - GET HOSTNAME COMPLETED
08:42:46 N/A      LISTEN EXECUTED
08:42:46 N/A      ACCEPT STARTED
08:42:46 N/A      BIND EXECUTED
08:42:46 N/A      GETCLIENTID EXECUTED - GET CLIENT ID COMPLETED
08:42:46 N/A      ACCEPT EXECUTED
08:42:46 N/A      GETSOCKNAME EXECUTED - GET SOCKET NAME COMPLETE
08:42:46 N/A      GETHOSTID EXECUTED - GET HOST ID COMPLETED
08:42:46 N/A      GETHOSTNAME EXECUTED - GET HOSTNAME COMPLETED
08:42:46 N/A      LISTEN EXECUTED - SOCK 0000 - LISTEN COMPLETED
08:42:46 N/A      ACCEPT STARTED - SOCK 0000 - ACCEPT INITIATED
08:42:46 N/A      ACCEPT EXECUTED - SOCK 0000 - AIO ACCEPT IN PROGRES
08:42:47 N/A      RESMGR detected termination of ACI internal service task
14:46:14 N/A      RESMGR detected termination of TSO task for Userid ADCDMST
14:47:23 N/A      RESMGR detected termination of TSO task for Userid ADCDMST
14:47:25 N/A      RESMGR detected termination of ISPF dialog task for Userid A
14:48:15 N/A      ENABLE VTB.AZKSMFT2
14:48:15 N/A      ENABLE VTB.AZKSMFT2 SECTION REXX-VTB.AZKSMFT2
14:48:15 N/A      AZK3900T RULE VTB.AZKSMFT2 FOR VTB MODIFYTABLE.SMF_* NOW ENA
14:49:01 N/A      RESMGR detected termination of TSO task for Userid ADCDMST
*****
*****
***** BOTTOM OF MESSAGES *****
Command ==> _
Scroll ==> PAGE
  
```

This is your BEST FRIEND when things don't work as expected.

To get more information, move the cursor to the line you are interested in and press Enter.