Spooling Around with SAPI

Dave Danner

Summit Technical Services, Inc

david.danner@associates.dhs.gov

Share in New York
Session 2665

August 19, 2004

Abstract

Do you have a need to get at data on your JES spool? Maybe you need to archive the joblog of an important started task, read spool data from an application program, or send a report on the spool as an email attachment. Well, the SYSOUT Application Program Interface (SAPI) is the answer. In this presentation, the speaker will discuss his experiences with the development of SYSOUT Retrieval Services (SRS), a 'shareware' product that uses SAPI to select spool data based on a number of user-specified criteria. An overview of the external workings of SRS will be presented, followed by a look at the code required to write your own SAPI applications. The presentation will focus on JES2, but the concepts will be of value to users of both JESes.

Agenda

- Introduction
  - Surfing Your JES Spool
  - Problems Getting at Data on Your JES Spool
  - SAPI Overview
- Part I - Sample SAPI Application
  - SRS Overview
  - Selecting SYSOUT
  - Specifying Control Options
  - Writing Your Own De-spooler Program
- Part II - Writing Your Own SAPI Programs
  - SAPI Programming Basics
  - Making the SSI Call
  - Processing the Data Returned
  - SAPI Program Logic Flow

Introduction

Surfing your JES Spool

Ever Have a Need to…?

- Archive a copy of spool data to DASD for future reference
  - JES JOBLOG or other SYSOUT files from important STC’s or batch jobs
  - Output from software install/maintenance jobs
    - ServerPac installs
    - SMP/E service APPLY & ACCEPT
- e-mail a report instead of printing it

Surfing your JES Spool

Ever Have a Need to…?

- Support file transfer via NJE
  - Application on Node A sends application on Node B a data file as SYSOUT
- Exploit modern data transfer technologies with legacy applications
  - Example: Application generates spool file for RJE. Want to replace the RJE with:
    - Transfer via FTP
    - Send as e-mail attachment
    - Write to MQSeries queue

Surfing your JES Spool

Ever Have a Need to…?

- Copy JES2 $TRCLOG files to DASD
  - Order files were written to spool usually needs to be preserved

The Problem…

There’s no easy way to get at data on the JES Spool

- SDSF PRINT-to-data set function
  - Good for making one-time copies of a single job from ISPF, but…
  - Multiple jobs require multiple print actions
  - Hard to invoke in batch or automate for repetitive use
- External Writers (XWTR)
  - Hard to automate
  - Usually one XWTR proc per system
    - Shared for all users!
  - Very limited SYSOUT selection criteria
    - Can’t select held output

The Problem…

There’s no easy way to get at data on the JES Spool

- Process SYSOUT (PSO) applications
  - Functionally stabilized
  - Supports only one SYSOUT request per address space
  - Limited SYSOUT selection criteria
- TSO TRANSMIT/RECEIVE
  - Requires userid on receiving end
  - Hard to automate

SAPI Overview

- *SYSOUT Application Program Interface*
- Introduced in OS/390 1.3 (1997)
  - SHARE members contributed to design
  - Intended to enhance/replace PSO
- Provides a programmable interface to output on the JES spool
  - Can request SYSOUT matching a variety of criteria
  - Each SYSOUT request is a SAPI “thread”
    - Can have multiple open threads *per task*
- Overall design is similar to PSO
  - Invoked via Sub System Interface (SSI)
  - Specify SYSOUT criteria (“filters”) on SSI call
  - Assembly language required

SAPI Overview

- Three Primary Functions:
  - PUT/GET – access spool data sets
  - Count – Scan output elements; return counts *
  - Bulk Modify – Mass alter of SYSOUT characteristics *

  * Not discussed in this presentation
- SAPI Limitations
  - Can only access non-busy spool data on the hardcopy queue
    - SYSOUT visible on SDSF HELD or OUTPUT displays
    - NOT busy on printer or other output device
  - Also consider using the SPOOL BROWSE function
    - See session 2664 *Exploiting New JES2 Interfaces*

Part I:
Sample SAPI Application

SYSOUT Retrieval Services (SRS) Overview

- Free "shareware" product
  - Current version is 1.2.0
- Uses SAPI to retrieve data from the JES spool based on user-supplied criteria
- Possible uses:
  - Use "out of the box" to write selected spool data to a file
  - Let SRS make SAPI calls; write your own de-spooler program to process spool data
  - Use as a sample SAPI application
- To get a copy:
  - Send an email to david.danner@associates.dhs.gov
  - Download from http://www.cbttape.org *(coming soon!)*

SYSOUT Retrieval Services (SRS) Overview

- To install:
  - Binary upload *Srclib* (source code) and *Linklib* (execution) files
  - Use TSO RECEIVE INDS(*dsn*) to convert to PDSes
  - Execution library must be APF authorized
  - Source library also contains documentation and samples

$ABOUT	contains info about SRS parameters and options
$INSTALL	contains detailed installation instructions
SRS	is a sample cataloged procedure to invoke SRS

SYSOUT Retrieval Services (SRS) Overview

- Originally designed to replace external writers
  - SRS defaults are similar to XWTR's
- Can execute as a STC or batch job
- Pass parameters on EXEC PARM=
- Two types of parameters
  - *SYSOUT Selection*
  - *SRS Control*

SRS SYSOUT Selection Parameters

- Specify the characteristics of SYSOUT to select
- These are either:
  - Passed directly to SAPI or
  - Used to set selection flags in the SAPI SYSOUT selection control block

SRS SYSOUT Selection Parameters

Q=*class*

- Specifies the SYSOUT class
- Only a single class can be specified
- Default: none

SRS SYSOUT Selection Parameters

J=*jobname*

- Specifies the jobname
- Duplicate jobnames can be selected
- Supports wildcard matching characters * (for multiple characters) and ? (for a single character)
- Default: none

SRS SYSOUT Selection Parameters

JI=*jobid*

- Specifies the 8-character JES jobid
- Supports all jobid formats:

J12345	JO12345	JOB12345
S12345	ST12345	STC12345
T12345	TS12345	TSU12345

- Supports large job numbers (up to 999,999)
- Default: none

SRS SYSOUT Selection Parameters

D=*dest*

- Specifies the destination ID
- Supports wildcard matching characters
- D=ALL specifies destination is not to be checked for selection
- Default: *LOCAL*

SRS SYSOUT Selection Parameters

F=*forms*

- Specifies the forms ID
- Supports wildcard matching characters
- Default: *STD*

SRS SYSOUT Selection Parameters

W=*writer*

- Specifies the writer ID
- Supports wildcard matching characters
- Default: null (SYSOUT with **no** writer name)

SRS SYSOUT Selection Parameters

STATUS=*stat*

- Specifies the status of the output
  - STATUS=HELD selects only OUTDISP=HOLD output
  - STATUS=NONHELD selects only OUTDISP=WRITE output
  - STATUS=ALL selects both OUTDISP=HOLD and WRITE output
- Default: *NONHELD*

SRS SYSOUT Selection Parameters

DDNAME=*ddn*

- Specifies the DDNAME of the spool data set
- Supports wildcard matching characters
- May cause only part of an output group to be selected
- Default: none

SRS Control Parameters

- Specify how SRS should operate
- Used internally by SRS and not passed to SAPI

SRS Control Parameters

LIM=*limit*

- Specifies the number of output groups to select
  - Multiple SYSOUT datasets grouped together in the same output group count as ONE
- Default: none (all matching output groups are selected)

SRS Control Parameters

WAIT=YES | NO

- Specifies whether SRS should wait for SYSOUT or end when no SYSOUT remains to be selected
  - If WAIT=YES, to terminate SRS issue:
    - STOP
    - MODIFY *srs*,STOP
    - MODIFY *srs*,SHUTDOWN
- Default: *NO*, if SRS is invoked in batch; *YES* otherwise

SRS Control Parameters

DISP=KEEP | DELETE

- Specifies whether selected SYSOUT should be kept or deleted from the JES spool after successful processing
- Default: *DELETE*

SRS Control Parameters

P=*program_name*

- Specifies the name of the program that SRS will call to process de-spooled data
  - SRS will issue a *LOAD* for this module
    - Must be accessible via STEPLIB, LNKLST, LPA, etc.
- Default: *SRSGENER* (included in the SRS LOADLIB)

SRS Control Parameters

PPARM=*program_parms*

- Specifies up to 80 bytes of data to be passed to the de-spooling program
- Parms are not syntax-checked or processed in any way by SRS
- Default: none

SRS Parameters

- To avoid inadvertently de-spooling everything in the output queue, at least *one* of the following must be specified:
  - Q=*class*
  - D=*dest*
  - J=*jobname*
  - F=*forms*
  - JI=*jobid*
  - W=*writer*

Default De-Spooling Program (SRSGENER)

- Writes data selected to an output file defined by the SYSUT2 DD statement
  - Can override via PPARM=OUTFILE=*ddname*
- If output file has no DCB attributes, SRS uses RECFM=FB with LRECL set to match the length of the first spool data set processed
- If more than one output group is selected (LIM>1), spool data sets are concatenated together into one logical sequential file
  - No separators between data sets or output groups

Writing Your Own De-spooling Program

- Specify via P=*program_name* control parm
- SRS will call (branch enter) program at:
  - Task initialization
  - Task termination
  - New data set selected
  - Process each data record
  - End-of-data set
- SRS will pass:
  - Address of SSOB (from SAPI)
  - Address and length of data record
  - PPARM=*program_parms* specified by user (if any)

SRS Example #1

- Write all spool files assigned a writer name of WTR1 to the SYSUT2 DD:

```
//DESPOOL  EXEC SRS,PARM='W=WTR1'
```

```
SRS500I SYSOUT Retrieval Services - Release 1.2.0
SRS505I WTR01 STAT=STARTING D=LOCAL W=WTR1 S=NONHELD
SRSG01I Writing spool data to DDNAME=SYSUT2   DSN=DWD.WTR1.OFFLOAD
SRS001I DWDJOB2(J0547707) Q=A D=LOCAL O=1.1.1 DS=1/JESMSGLG on WTR01 (24 RECS)
SRS001I DWDJOB2(J0547707) Q=A D=LOCAL O=1.1.1 DS=2/JESJCL on WTR01 (6 RECS)
SRS001I DWDJOB2(J0547707) Q=A D=LOCAL O=1.1.1 DS=3/JESYSMSG on WTR01 (11 RECS)
SRS001I AUTOCMDS(S0547798) Q=A D=LOCAL O=2.1.1 DS=1/SYSPRINT on WTR01 (390 RECS)
SRS001I ILBR1UP(S0547868) Q=Z D=LOCAL O=1.1.1 DS=1/JESMSGLG on WTR01 (24 RECS)
SRS001I ILBR1UP(S0547868) Q=Z D=LOCAL O=1.1.1 DS=2/JESJCL on WTR01 (50 RECS)
SRS001I ILBR1UP(S0547868) Q=Z D=LOCAL O=1.1.1 DS=3/JESYSMSG on WTR01 (53 RECS)
SRS001I SCHRESCK(S0548218) Q=X D=LOCAL O=1.1.1 DS=1/JESMSGLG on WTR01 (21 RECS)
SRS001I SCHRESCK(S0548218) Q=X D=LOCAL O=1.1.1 DS=2/JESJCL on WTR01 (46 RECS)
SRS001I SCHRESCK(S0548218) Q=X D=LOCAL O=1.1.1 DS=3/JESYSMSG on WTR01 (48 RECS)
SRS001I SCHRESCK(S0548218) Q=X D=LOCAL O=1.1.1 DS=4/SYSTSPRT on WTR01 (137 RECS)
SRS022I No more SYSOUT datasets to select
```

SRS Example #2

- Write a copy of the JES data sets for job DBPXMUS to DD TAPELOG

```
//DESPOOL  EXEC SRS,
//     PARM='J=DBPXMUS,DDNAME=JES*,PPARM=OUTFILE=TAPELOG,DISP=KEEP,LIM=1'
```

```
SRS500I SYSOUT Retrieval Services - Release 1.2.0
SRS505I WTR01 STAT=STARTING J=DBPXMUS D=LOCAL DDNAME=JES* S=NONHELD DISP=KEEP
SRSG01I Writing spool data to DDNAME=TAPELOG  DSN=DBA.DBPXMUS.JESLOGS.G1462V00
SRS001I DBPXMUS(STC05569) Q=2 D=LOCAL O=2.1.1 DS=1/JESMSGLG on WTR01 (199,649
   RECS)
SRS001I DBPXMUS(STC05569) Q=2 D=LOCAL O=2.1.1 DS=2/JESJCL on WTR01 (100 RECS)
SRS001I DBPXMUS(STC05569) Q=2 D=LOCAL O=2.1.1 DS=3/JESYSMSG on WTR01 (18,558
   RECS)
SRS021I SYSOUT retrieval limit reached
```

SRS Example #3

- Start SRS as a started task
  - When output with DEST=SRSTEST is written to spool, call MYSRSPGM to process the spool data set

```
START SRS,PARM='D=SRSTEST,P=MYSRSPGM'
04223 15:25:50.45 IEF403I SRS - STARTED - TIME=15.25.50
04223 15:25:50.47 SRS500I SYSOUT Retrieval Services - Release 1.2.0
04223 15:25:50.47 SRS501I WTR01 attached
04223 15:25:50.47 SRS505I WTR01 STAT=STARTING P=MYSRSPGM D=SRSTEST S=NONHELD
04223 15:25:50.48 SRS003I WTR01 Initialization complete
04223 16:48:15.33 SRS001I SCHRESCK(S0106872) Q=Q D=SRSTEST O=1.1.1 DS=1/SYSTSPRT
                  on WTR01 (137 RECS)
04223 18:10:44.12 SRS001I ABNDEXIT(S0100564) Q=D D=SRSTEST O=1.1.1 DS=1/SYSPRINT
                  on WTR01 (21 RECS)
STOP SRS
04223 20:01:05.22 SRS502I STOP command accepted
04223 20:01:05.22 SRS012I WTR01 SAPI interface deactivated
04223 20:01:05.22 SRS013I WTR01 SHUTDOWN complete
04223 20:01:05.26 IEF404I SRS - ENDED - TIME=20.46.17
```

SRS Return codes

0: *Normal processing - at least one SYSOUT data set was processed*

4: *Normal processing - no SYSOUT data sets were found*

8: *Error in input parameter list*

16: *De-spooling program requested termination*

20: *SAPI SSI call failed*

24: *SRS SAPI interface subtask abended*

32: *De-spooling program abended*

Ideas for Future SRS Enhancements

- Support multiple de-spooling tasks under the same address space
- Allow installation to specify defaults
- Add separator records between data sets or output groups
- Return info from the SWB
- Support variable record length output files
- Support *all* selection criteria that is supported by SAPI, e.g.
    - Multiple SYSOUT classes
    - Selection by size (# lines) of file

Part II:
Writing Your Own SAPI Programs

SAPI Programming Basics

- To use SAPI PUT/GET mode, you'll need to:
  - Call SAPI to obtain a data set on spool that matches your selection criteria
  - Use the data set name and token returned by SAPI to dynamically allocate the spool data set
  - OPEN, READ, and CLOSE the spool data set
  - Dynamically un-allocate the data set
  - Call SAPI again to:
    - Specify the disposition of the last data set
    - Get the next data set (optional)
- SRS module SRSTASK is a good sample
- See *Using the Subsystem Interface* (SA22-7642)

Making the SAPI Call

- SAPI is called via the Subsystem Interface (SSI)
- Before making the call we need to build:
  - Subsystem Options Block (SSOB)
  - Subsystem Identification Block (SSIB) – optional
  - SSOB extension unique for SAPI
- The IEFSSREC macro invokes the SSI
- Supervisor State required!

Making the SAPI Call

- Subsystem Options Block (SSOB)
  - Required for all SSI calls
  - Mapped by macro IEFJSSOB
  - A code identifies the requested function (SSOBFUNC)
    - 79 (decimal) for SAPI
- Subsystem Identification Block (SSIB)
  - Identifies which subsystem will service the request (SSIBSSNM)
  - Mapped by macro IEFJSSIB
  - Pointed to by SSOB (SSOBSSIB)
    - If SSOBSSIB=zero, the "life-of-job SSIB" is used
      - Created automatically by the system
      - Specifies the subsystem (JES) that initiated the currently running job
      - Use this unless you want to route the SAPI call to an alternate JES

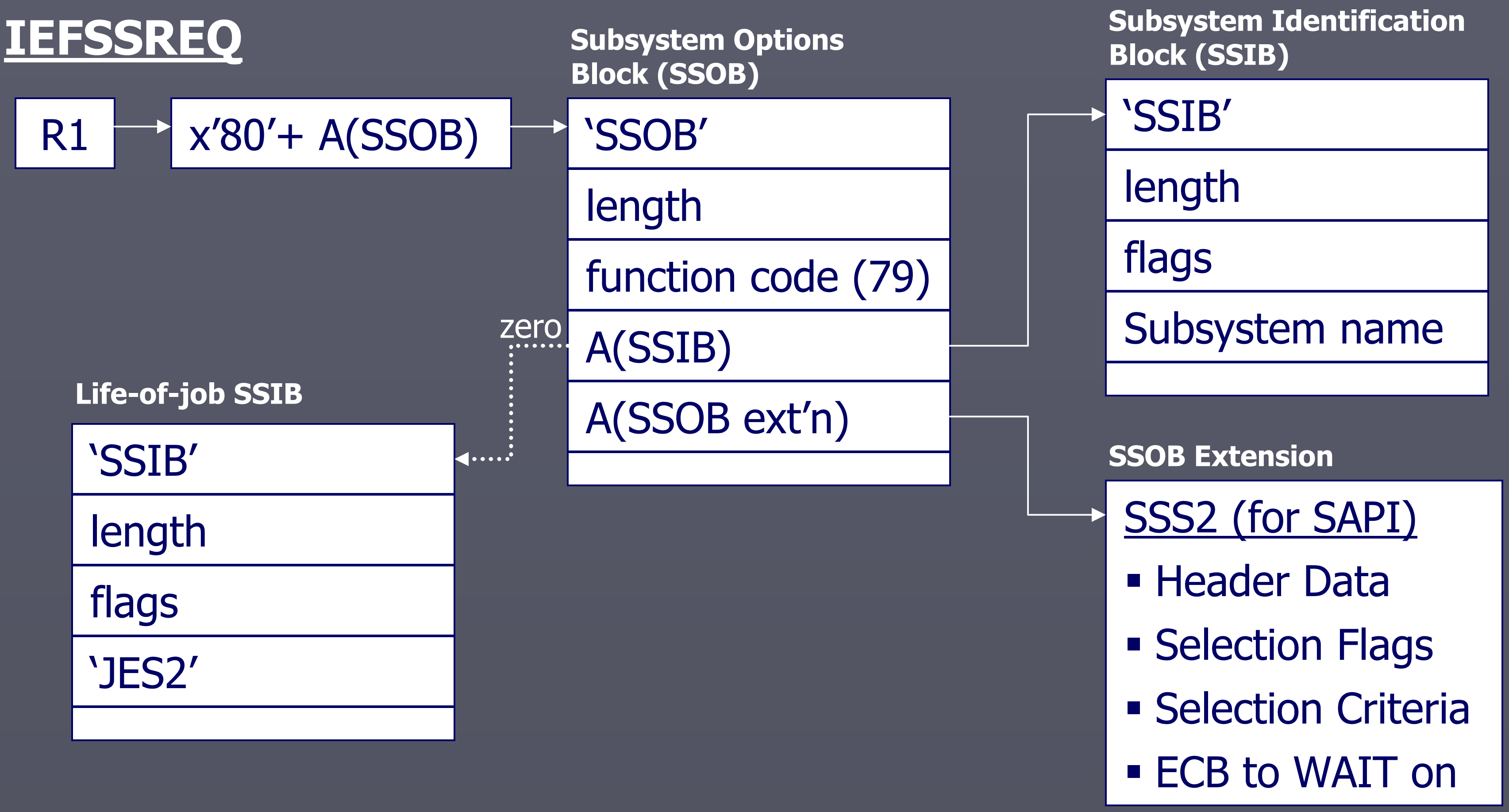
Making the SAPI Call

- SSOB Extension
  - Pointed to by SSOB (SSOBINDV)
  - Contains information specific to SAPI
    - Mapped by macro IAZSSS2 (all fields start with "SSS2")
    - Input fields are documented in section 3.1.8.18 "Input-Only Fields" of *Using the Subsystem Interface*
  - Tells SAPI what SYSOUT you want to select
    - Usually specified by a value/significance flag pair, for example:

```
MVC    SSS2JOBN,=CL8'MYJOB'   Set job name to select
OI     SSS2SEL1,SSS2SJBN      Set "check job name" flag
```

  - Points to an ECB SAPI will post when SYSOUT becomes available (if you want to wait)
  - Specifies the disposition of the previous data set returned
  - Tells SAPI that you want to terminate the thread

Making the SAPI Call



IEFSSREQ

R1 → x'80'+ A(SSOB)

Subsystem Options Block (SSOB)

- 'SSOB'
- length
- function code (79)
- A(SSIB)
- A(SSOB ext'n)

zero

Subsystem Identification Block (SSIB)

- 'SSIB'
- length
- flags
- Subsystem name

Life-of-job SSIB

- 'SSIB'
- length
- flags
- 'JES2'

SSOB Extension

SSS2 (for SAPI)

- Header Data
- Selection Flags
- Selection Criteria
- ECB to WAIT on

Making the SAPI Call

- After the call to IEFSSREQ, R15 contains a return code from the SSI
  - 0: *Successful completion - request went to the subsystem*
  - 4: *Subsystem does not support this function*
  - 8: *Subsystem exists, but is not active*
  - 12: *Subsystem is not defined to MVS*
  - 16: *Pointer to the SSOB or the SSIB or function code is not valid*
  - 20: *SSOB or SSIB have invalid lengths or formats*
  - 24: *SSI has not been initialized*
- Anything other than RC=0 is bad news!

Making the SAPI Call

- SSOBRETN in the SSOB contains a return code from SAPI

0: *Data set returned*
4: *No more data sets to select*
8: *Invalid search arguments*
12: *Unable to process now*
16: *Duplicate jobnames (SSS2SDUP was specified)*
20: *Invalid destination specified*
24: *Authorization failed*
28: *Token map failed*
32: *Logic error (See SSS2REAS)*
36: *SSS2CLAS not A-Z and not 0-9*
40: *Disposition settings incorrect (See SSS2REAS)*
44: *Disposition not uniform (See SSS2DSH)*

- RC = 0 and 4 should be expected
- RC= 12 means JES might be down; retry later

Processing the data returned

- If R15=0 *and* SSOBRETN=4 *(No more data sets to select),* SAPI processed the request successfully, but could not find any data sets that matched the selection criteria
- Here, you have two choices:
  - Wait on the ECB whose address you provided in the SSS2
    - JES will post this ECB when SYSOUT is available
    - Retry the SAPI call to return the data set
  - Terminate the SAPI thread
- NOTE: SAPI caller must have SAF UPDATE authority to JESSPOOL *nodename.spooldsn* for each spool data set
  - If SAF check fails, SAPI sets SSOBRETN=4
    - JES2 issues $HASP186 to SYSLOG
      - Set DEBUG SECURITY=YES to see security msgs

Processing the data returned

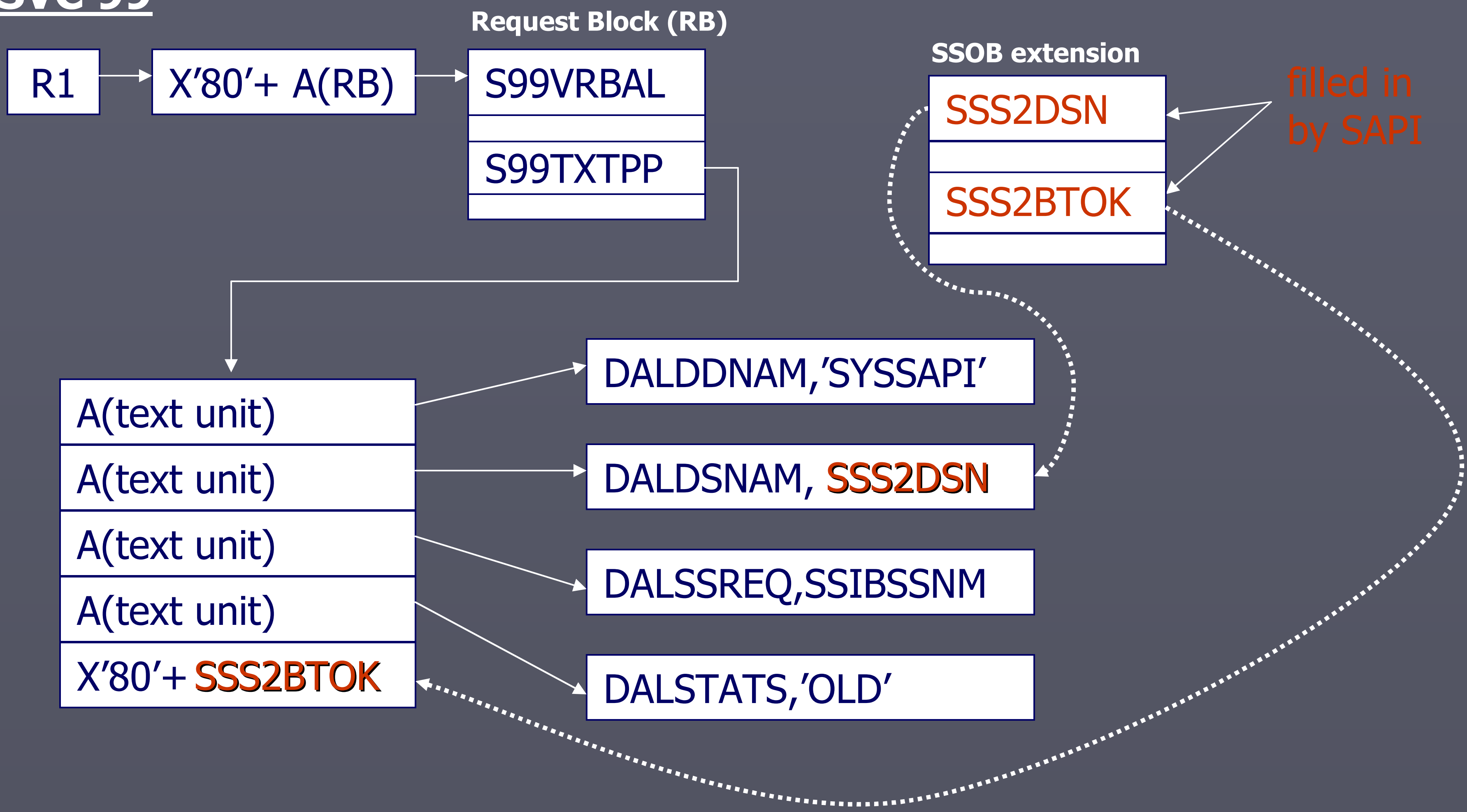
- If R15=0 *and* SSOBRETN=0, SAPI has found a data set that matches the specified selection criteria
- SAPI returns *one* data set (PDDB for JES2) at a time
  - Flags indicate when data set is first/last in output group (JOE for JES2)
  - You can choose to delete some data sets and keep others *even within the same output group*
- SAPI fills in fields in the SSS2 to describe the data set
  - See section 3.1.8.22 "Output-Only Fields" of *Using the Subsystem Interface* for info returned
  - Some fields will be needed to dynamically allocate the data set:
    - SSS2DSN = the spool data set name
    - SSS2BTOK = allocation text unit supplied by SAPI

Processing the data returned

- Before reading the data set, we first need to dynamically allocate it (SVC 99) using the following text units:
    - DDNAME = *ddname*
        - Must match the DDNAME in the DCB that will be used to OPEN the data set
    - DSN = SSS2DSN (from SSS2)
    - SUBSYS = *ssn*
        - Must be same subsystem (from SSIB) that serviced SAPI call
        - Use SSIBSSNM from life-of-job SSIB if SSOBSSIB=0
            - PSA → TCB → JSCB → life-of-job SSIB
    - DISP = "OLD"
    - SSS2BTOK (raw text unit from SSS2)

Processing the data returned

SVC 99



Request Block (RB)

R1

X’80’+ A(RB)

S99VRBAL

S99TXTPP

SSOB extension

SSS2DSN

SSS2BTOK

filled in by SAPI

A(text unit)

A(text unit)

A(text unit)

A(text unit)

X’80’+ SSS2BTOK

DALDDNAM,’SYSSAPI’

DALDSNAM, SSS2DSN

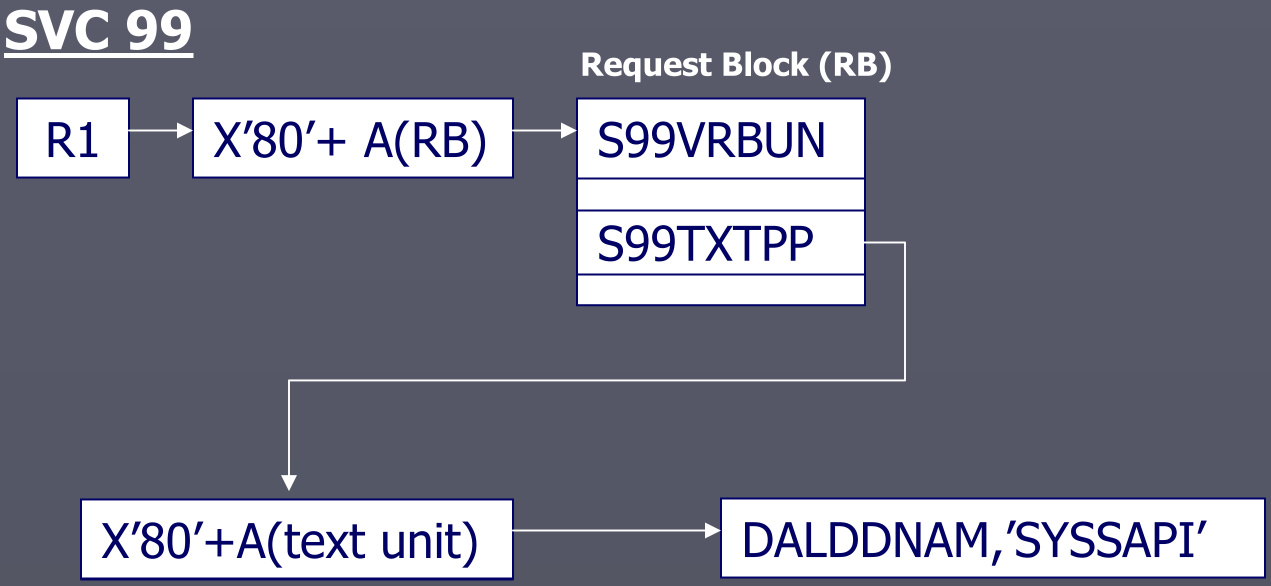
DALSSREQ,SSIBSSNM

DALSTATS,’OLD’

Processing the data returned

- After the file has been dynamically allocated, it can be processed like any other sequential data set:
  - OPEN with DCB that specifies the same DDNAME used on DYNALLOC
  - GET (QSAM) each record
  - CLOSE when finished (EOF)
- Un-allocate the file when finished
  - We only need the DDNAME for this!

Processing the data returned



SVC 99

Request Block (RB)

R1

X’80’+ A(RB)

S99VRBUN

S99TXTPP

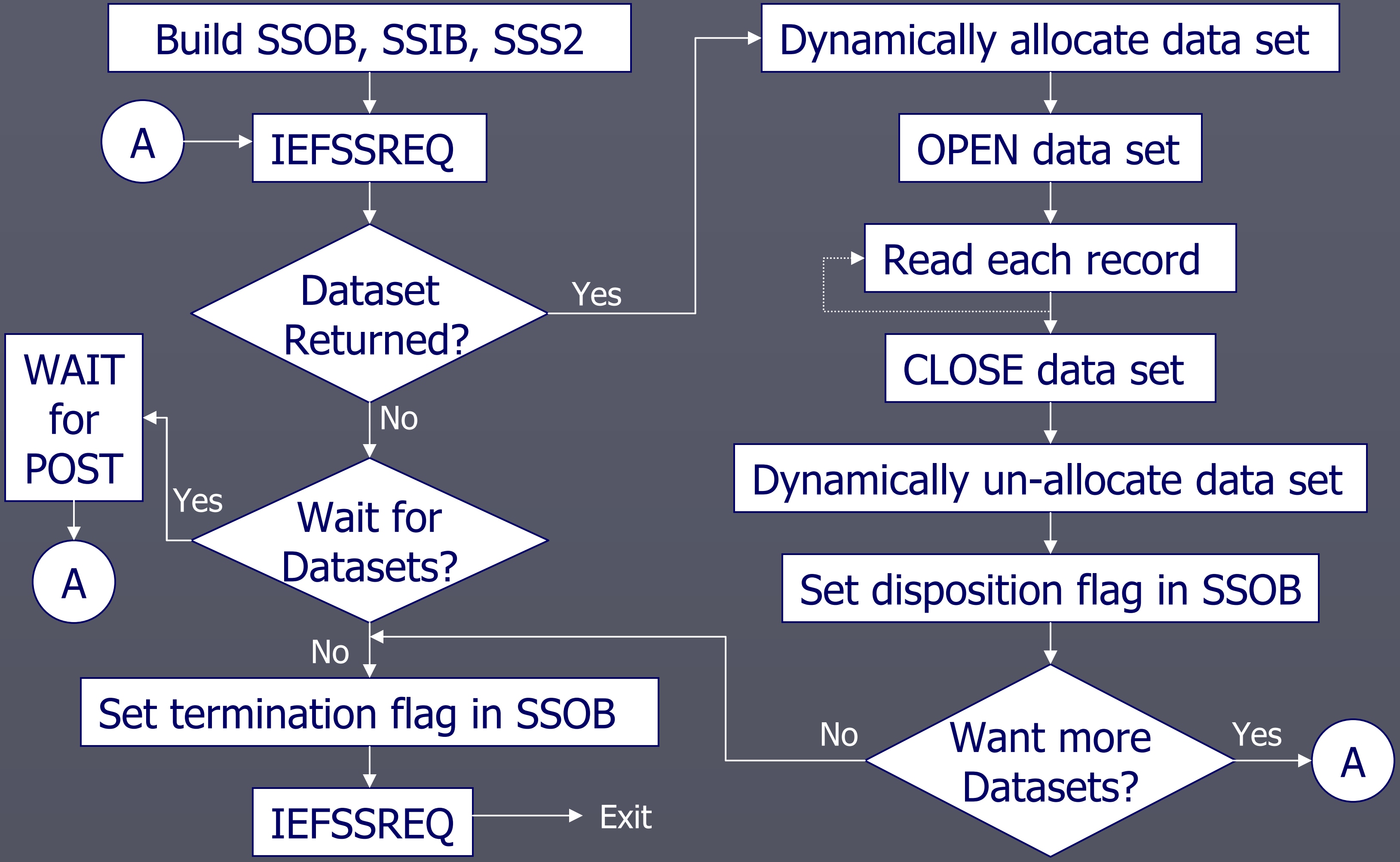
X’80’+A(text unit)

DALDDNAM,’SYSSAPI’

Processing the data returned

- After the data set has been un-allocated, we need to call SAPI again to specify the disposition of the data set returned on the *prior* call
  - Set SSS2DKPE in SSS2DSP1 to KEEP the data set
    - Otherwise, the data set is DELETED
    - Other flags can make the data set held, non-selectable, etc.
- The same call tells SAPI to either return another data set or terminate the SAPI thread
  - Set SSS2CTRL in SSS2MSC1 to terminate the thread
    - Otherwise, SAPI assumes you want another data set

SAPI Program Logic Flow



Build SSOB, SSIB, SSS2

A

IEFSSREQ

Dataset Returned?

Yes

No

WAIT for POST

A

Yes

Wait for Datasets?

No

Set termination flag in SSOB

IEFSSREQ

Exit

Dynamically allocate data set

OPEN data set

Read each record

CLOSE data set

Dynamically un-allocate data set

Set disposition flag in SSOB

No

Want more Datasets?

Yes

A

Bibliography

- *z/OS MVS Using the Subsystem Interface* (SA22-7642)

  IBM Corporation

- *Exploiting New JES2 Interfaces*

  SHARE, Winter 2004

  Tom Wasik