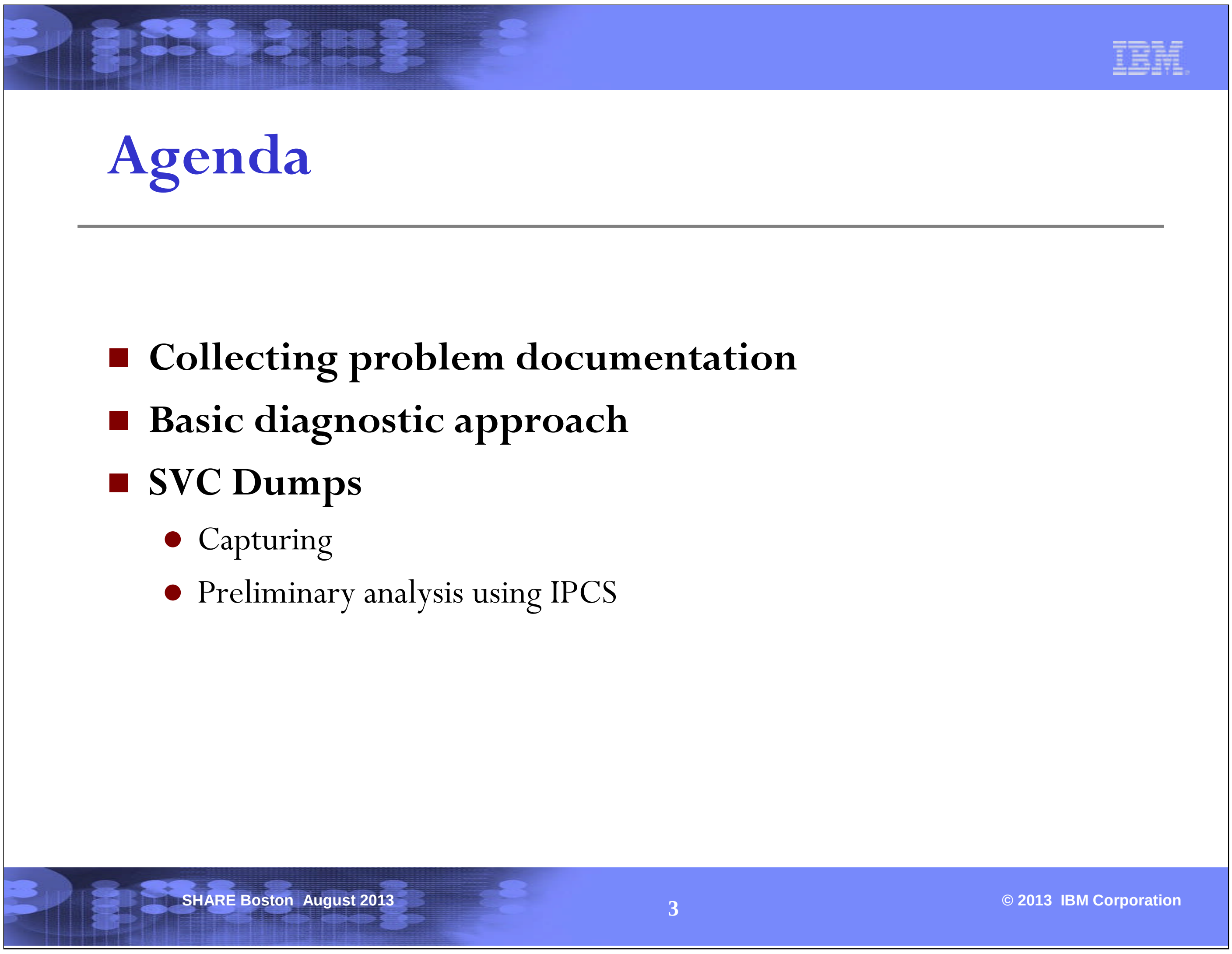
Session 14181
z/OS Debugging: *Old Tricks* *for* *New Dogs*

zNextGen Project – August 14th, 2013

Jerry Ng jerryng@us.ibm.com
Patty Little plittle@us.ibm.com
IBM Poughkeepsie

SHARE Boston August 2013

Trademarks

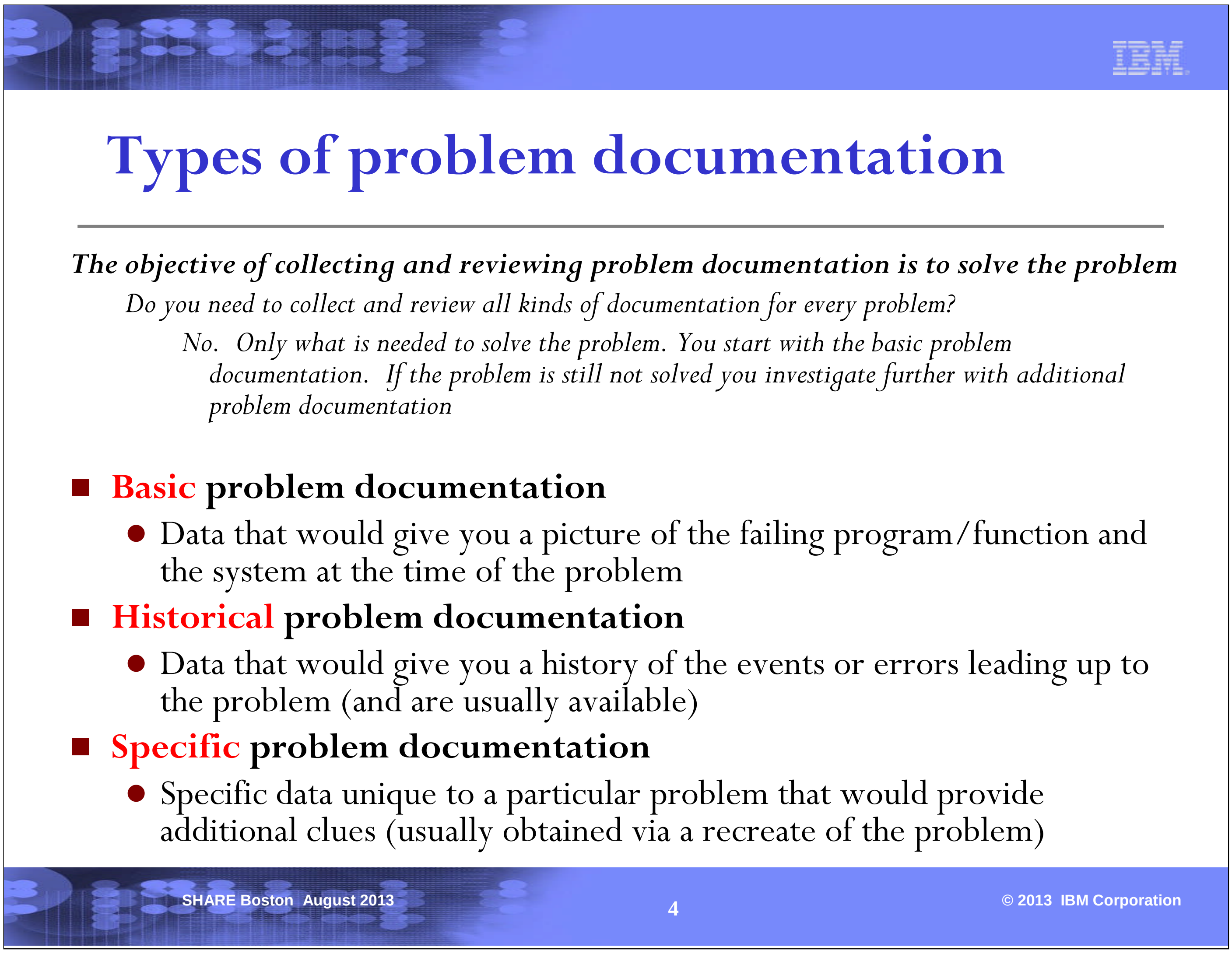


Agenda

- Collecting problem documentation
- Basic diagnostic approach
- SVC Dumps
  - Capturing
  - Preliminary analysis using IPCS

This presentation will discuss:

-the various kinds of problem documentation available in z/OS debugging
-basic diagnostic approaches
-the preliminary steps of diagnosing an SVC dump.



Types of problem documentation

The objective of collecting and reviewing problem documentation is to solve the problem

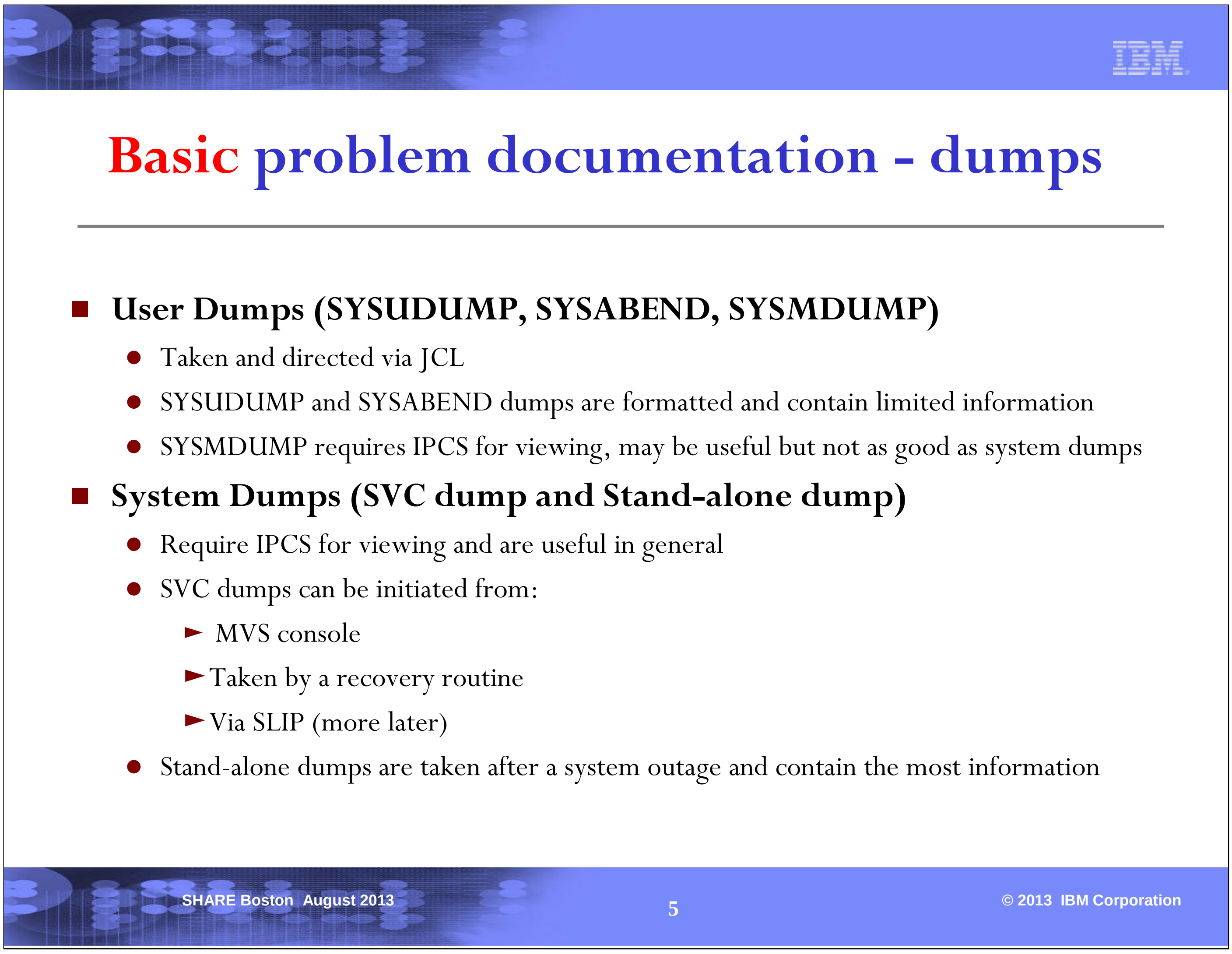
Do you need to collect and review all kinds of documentation for every problem?

No. Only what is needed to solve the problem. You start with the basic problem documentation. If the problem is still not solved you investigate further with additional problem documentation

- **Basic problem documentation**
  - Data that would give you a picture of the failing program/function and the system at the time of the problem
- **Historical problem documentation**
  - Data that would give you a history of the events or errors leading up to the problem (and are usually available)
- **Specific problem documentation**
  - Specific data unique to a particular problem that would provide additional clues (usually obtained via a recreate of the problem)

Even though there are different kinds of problem documentation, the debugger may not need to review all of them to resolve a problem. The key thing to do is to follow a logical diagnostic approach and continue to review the documentation until the problem is resolved. Start with the basic problem documentation and then analyze additional documentation if necessary.

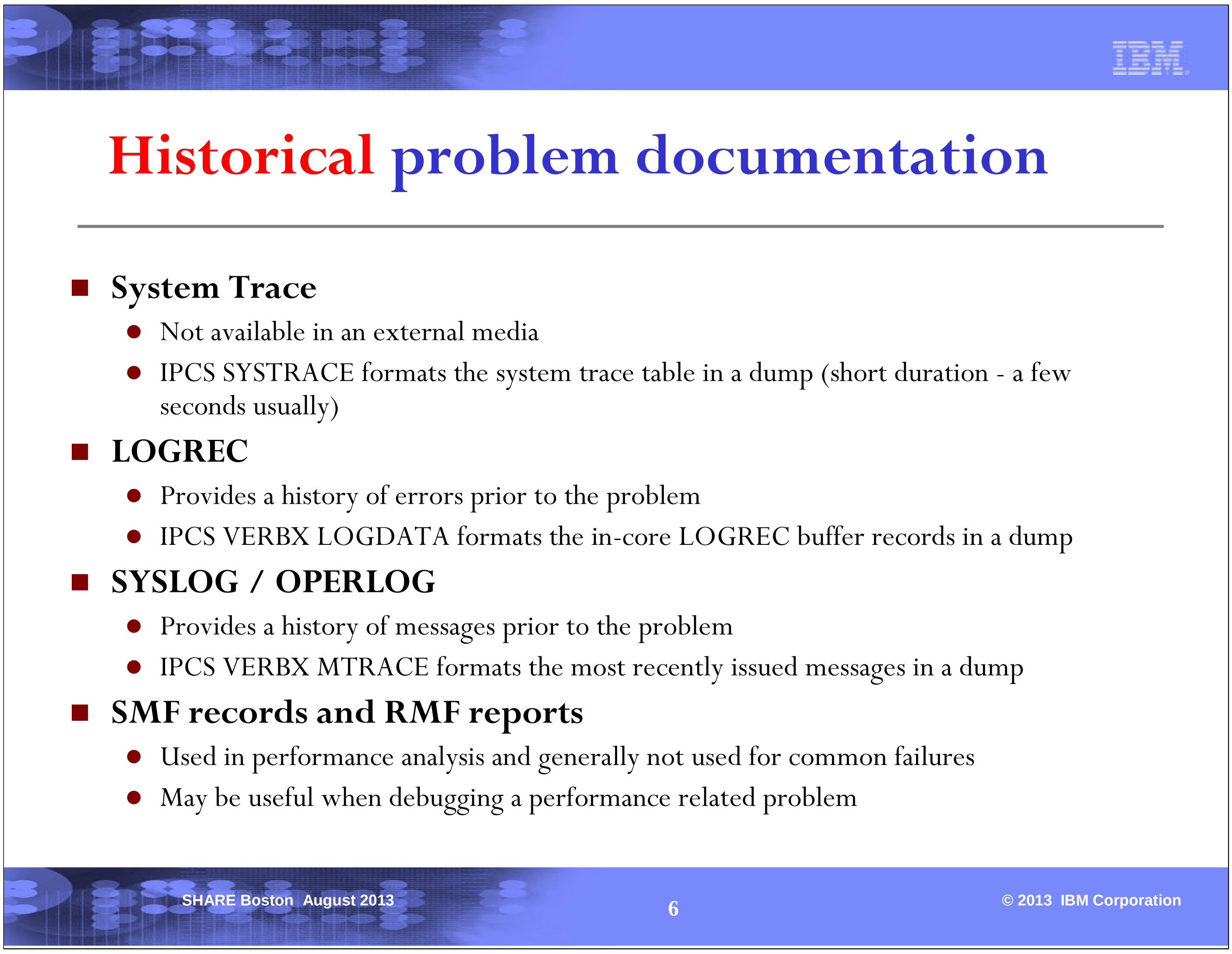
Sometimes the basic problem documentation may not be available and the debugger would have to use ways such as SLIP to gather them.



Basic problem documentation - dumps

- **User Dumps (SYSUDUMP, SYSABEND, SYSMDUMP)**
  - Taken and directed via JCL
  - SYSUDUMP and SYSABEND dumps are formatted and contain limited information
  - SYSMDUMP requires IPCS for viewing, may be useful but not as good as system dumps
- **System Dumps (SVC dump and Stand-alone dump)**
  - Require IPCS for viewing and are useful in general
  - SVC dumps can be initiated from:
    - ► MVS console
    - ► Taken by a recovery routine
    - ► Via SLIP (more later)
  - Stand-alone dumps are taken after a system outage and contain the most information

The basic problem documentation is a dump. Dumps give a picture of the failing program and the system at the time of the problem. We do not usually find user dumps very useful in debugging z/OS problems, so make sure you collect an SVC dump for a problem.

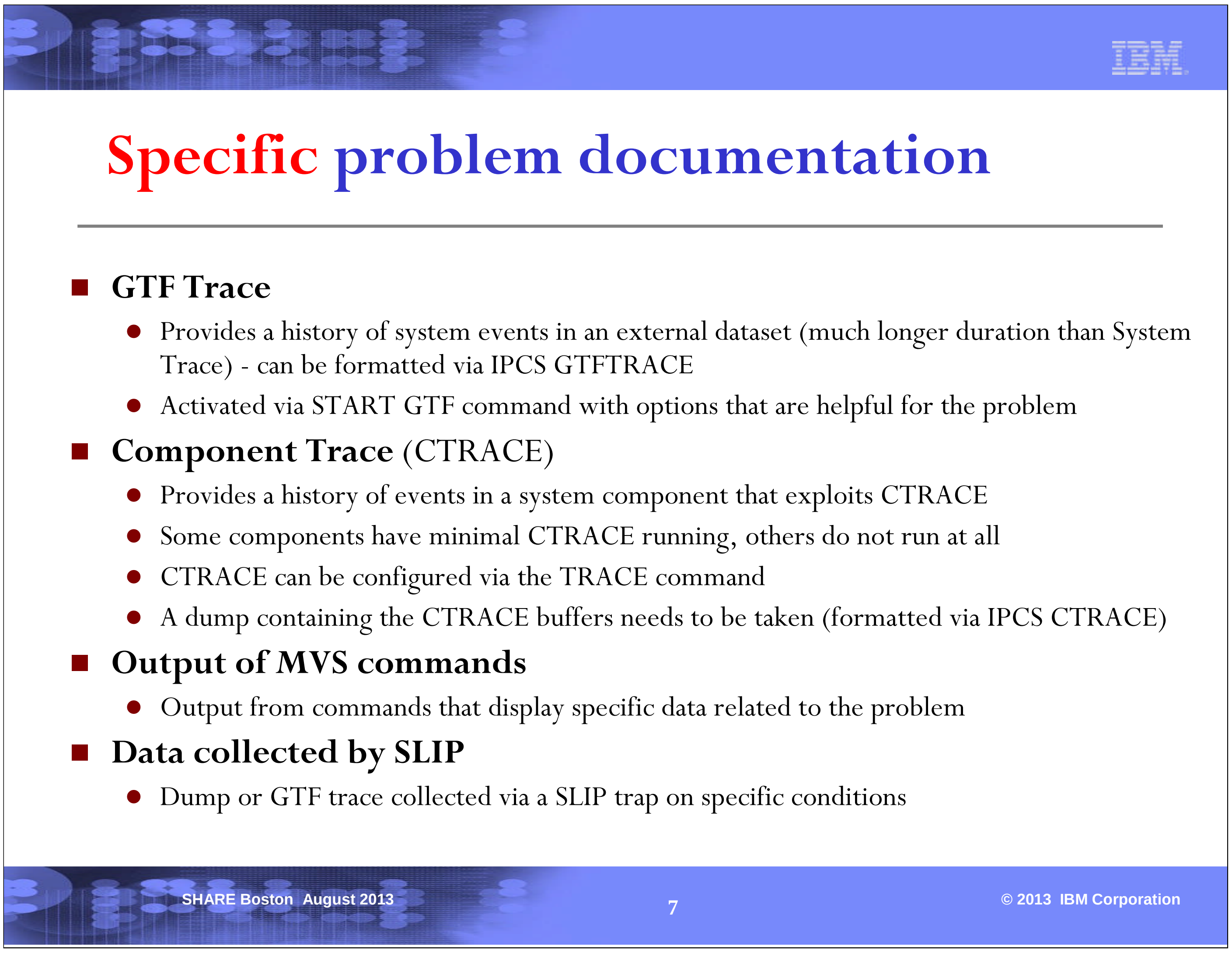


Historical problem documentation

- **System Trace**
  - Not available in an external media
  - IPCS SYSTRACE formats the system trace table in a dump (short duration - a few seconds usually)
- **LOGREC**
  - Provides a history of errors prior to the problem
  - IPCS VERBX LOGDATA formats the in-core LOGREC buffer records in a dump
- **SYSLOG / OPERLOG**
  - Provides a history of messages prior to the problem
  - IPCS VERBX MTRACE formats the most recently issued messages in a dump
- **SMF records and RMF reports**
  - Used in performance analysis and generally not used for common failures
  - May be useful when debugging a performance related problem

The historical problem documentation shows a history of errors or events leading up to the problem. They are usually available in a z/OS system as they are turned on and generated continuously.

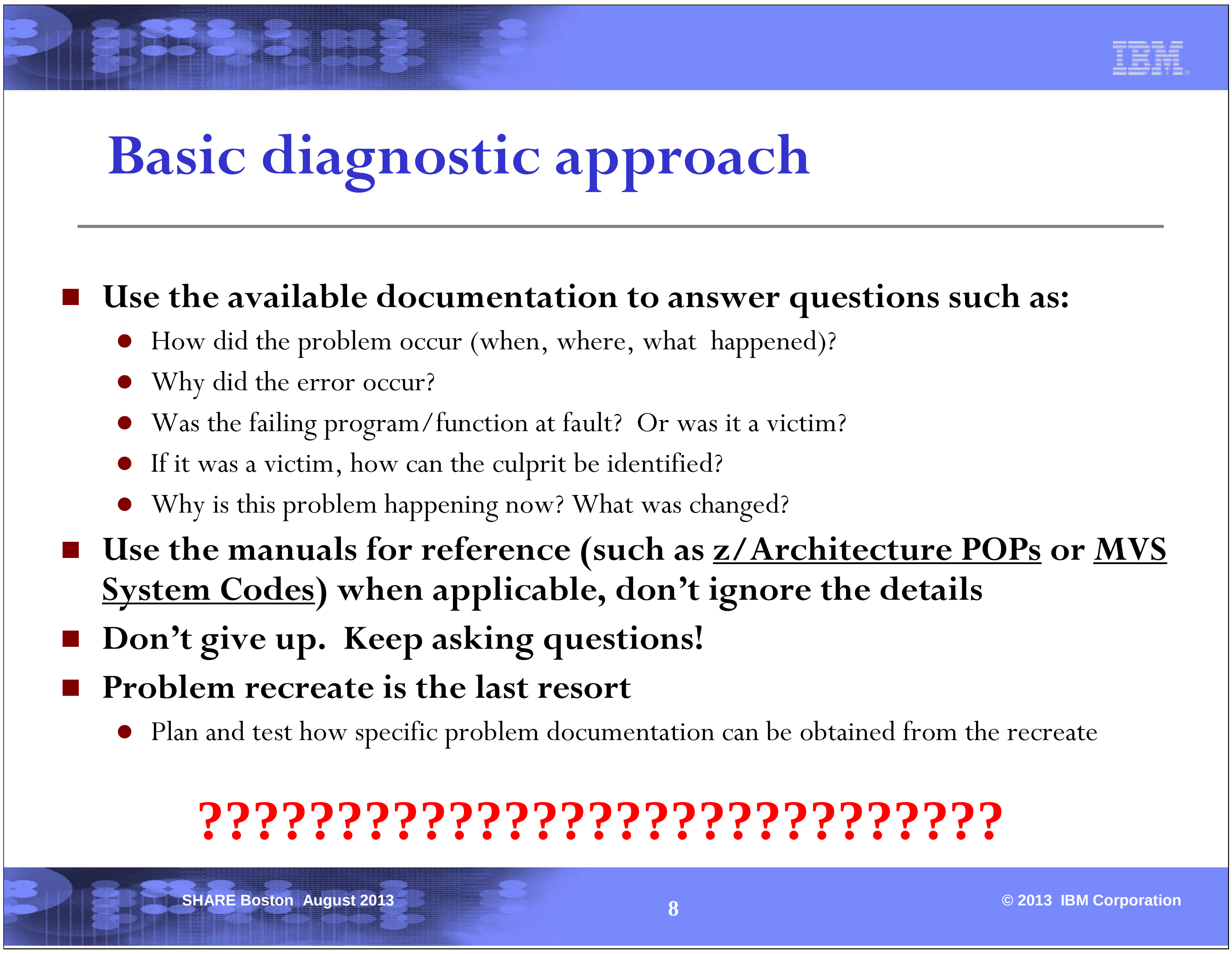
Note that SMF records and RMF reports are generally not useful for a common failure such as an abend. We would usually review the system trace first, followed by the LOGREC and SYSLOG.



Specific problem documentation

- **GTF Trace**
  - Provides a history of system events in an external dataset (much longer duration than System Trace) - can be formatted via IPCS GTFTRACE
  - Activated via START GTF command with options that are helpful for the problem
- **Component Trace** (CTRACE)
  - Provides a history of events in a system component that exploits CTRACE
  - Some components have minimal CTRACE running, others do not run at all
  - CTRACE can be configured via the TRACE command
  - A dump containing the CTRACE buffers needs to be taken (formatted via IPCS CTRACE)
- **Output of MVS commands**
  - Output from commands that display specific data related to the problem
- **Data collected by SLIP**
  - Dump or GTF trace collected via a SLIP trap on specific conditions

Specific problem documentation is usually collected via a recreate of the problem. The debugger should plan and test how to gather the specific problem documentation and the information that should be collected. Every attempt should be made to avoid having to recreate the problem over and over again.



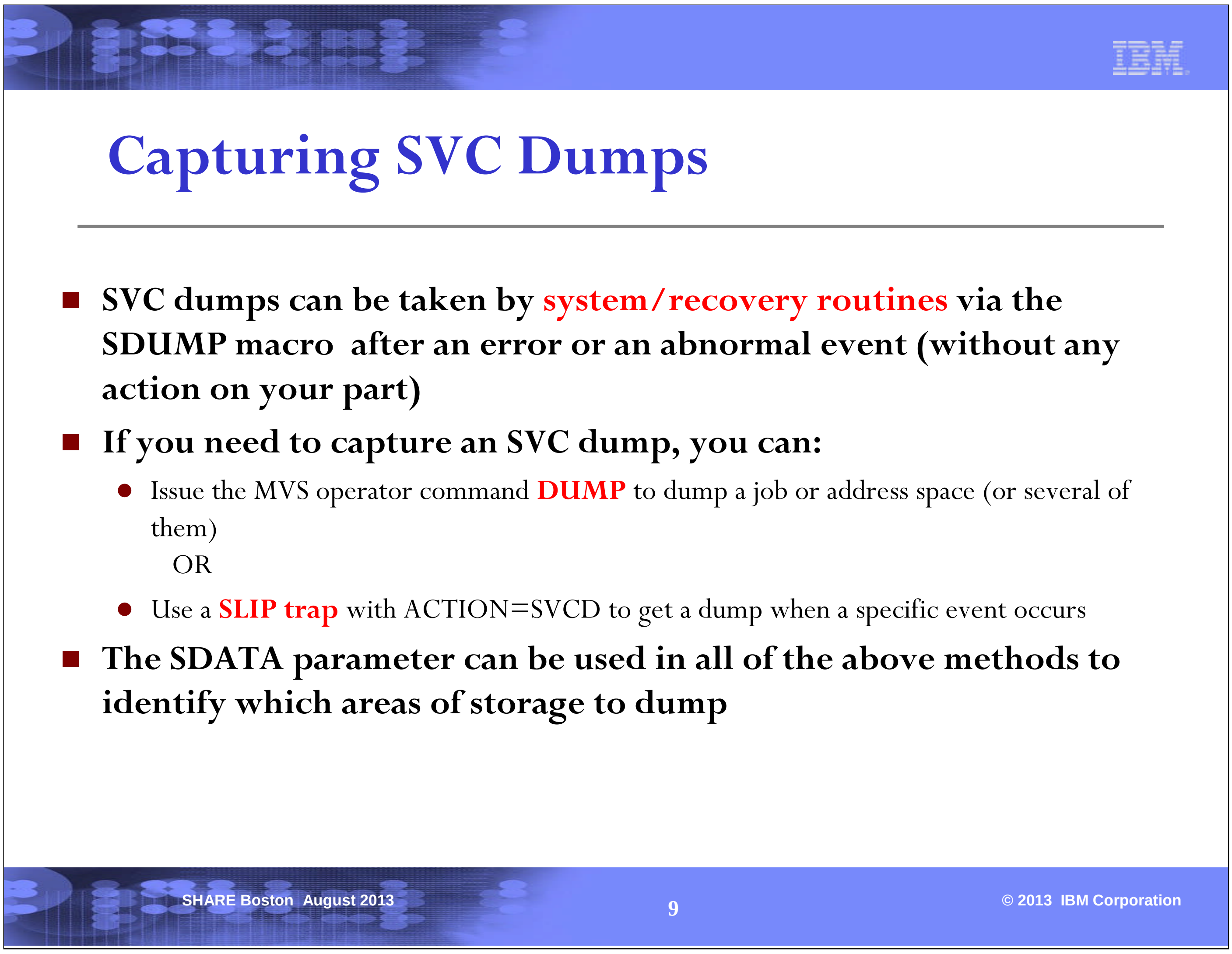
IBM

Basic diagnostic approach

- **Use the available documentation to answer questions such as:**
  - How did the problem occur (when, where, what happened)?
  - Why did the error occur?
  - Was the failing program/function at fault? Or was it a victim?
  - If it was a victim, how can the culprit be identified?
  - Why is this problem happening now? What was changed?
- **Use the manuals for reference (such as z/Architecture POPs or MVS System Codes) when applicable, don't ignore the details**
- **Don't give up. Keep asking questions!**
- **Problem recreate is the last resort**
  - Plan and test how specific problem documentation can be obtained from the recreate

?????????????????????????????

The basic diagnostic technique is to find out all the facts about the problem and then use a logical approach to solve the problem. Do not ignore the details or make assumptions without facts. Assume recreate is impossible and you have this one chance to solve the problem. Exhaust all avenues and turn over every stone.



Capturing SVC Dumps

- SVC dumps can be taken by system/recovery routines via the SDUMP macro after an error or an abnormal event (without any action on your part)
- If you need to capture an SVC dump, you can:
  - Issue the MVS operator command **DUMP** to dump a job or address space (or several of them)

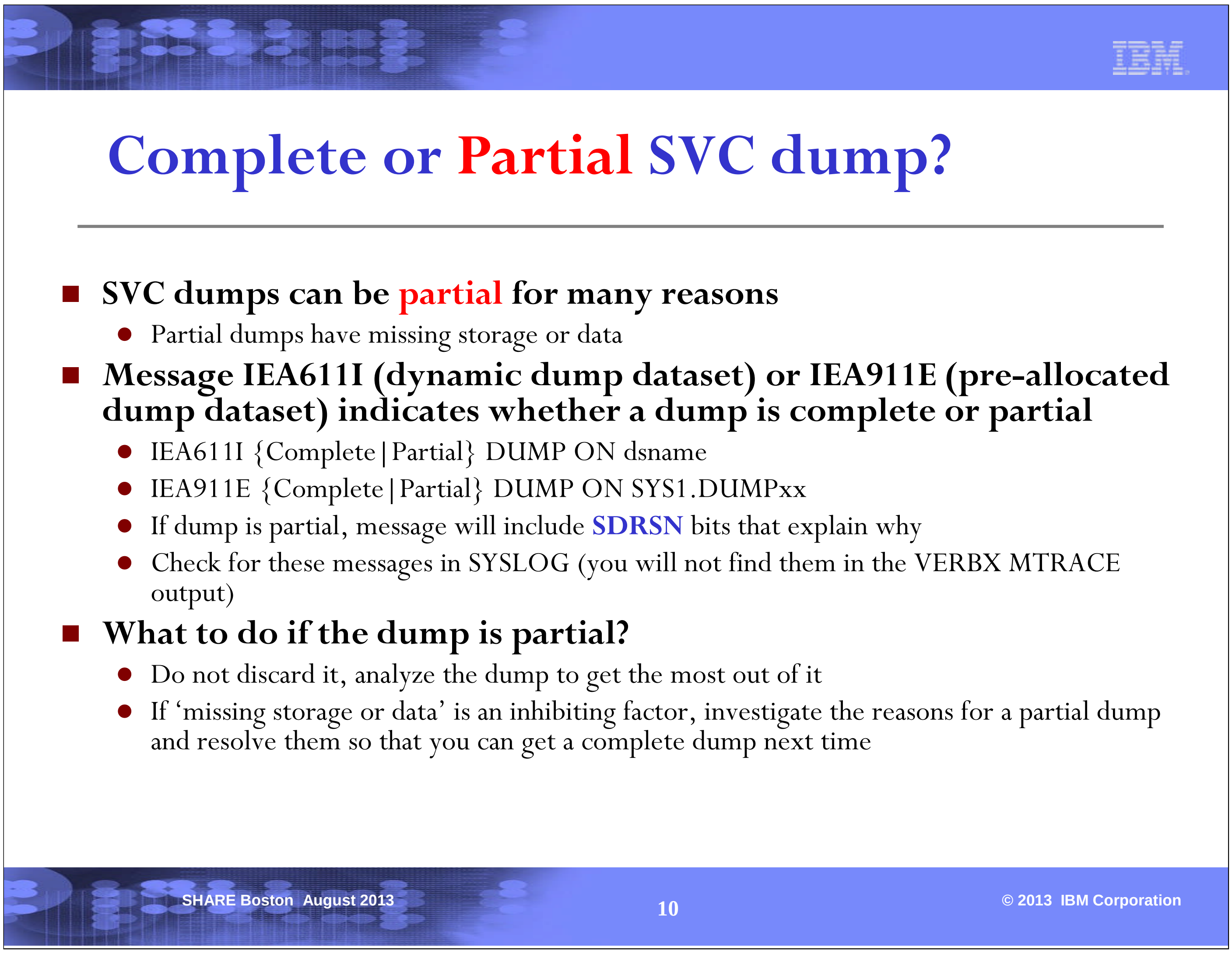
    OR
  - Use a **SLIP trap** with ACTION=SVCD to get a dump when a specific event occurs
- The SDATA parameter can be used in all of the above methods to identify which areas of storage to dump

There are 3 ways that SVC dumps can be taken:

•The SDUMP macro can be used by a recovery routine to take a dump. These dumps are usually the first sign of s problem.

•The DUMP command can be used to take a picture of an address space or a job, usually after they are hung.

•SLIP traps are very powerful in capturing a dump when an event occurs.



Complete or Partial SVC dump?

- **SVC dumps can be partial for many reasons**
  - Partial dumps have missing storage or data
- **Message IEA611I (dynamic dump dataset) or IEA911E (pre-allocated dump dataset) indicates whether a dump is complete or partial**
  - IEA611I {Complete|Partial} DUMP ON dsname
  - IEA911E {Complete|Partial} DUMP ON SYS1.DUMPxx
  - If dump is partial, message will include **SDRSN** bits that explain why
  - Check for these messages in SYSLOG (you will not find them in the VERBX MTRACE output)
- **What to do if the dump is partial?**
  - Do not discard it, analyze the dump to get the most out of it
  - If 'missing storage or data' is an inhibiting factor, investigate the reasons for a partial dump and resolve them so that you can get a complete dump next time

An SVC dump can be complete or partial. A partial dump will have missing storage or data, but it can still be useful. It is important to check if the dump is partial if you find that you are unable to display some storage or data from the dump.

The SDRSN is mapped in MVS Data Areas. The most common reasons for a partial dump are:

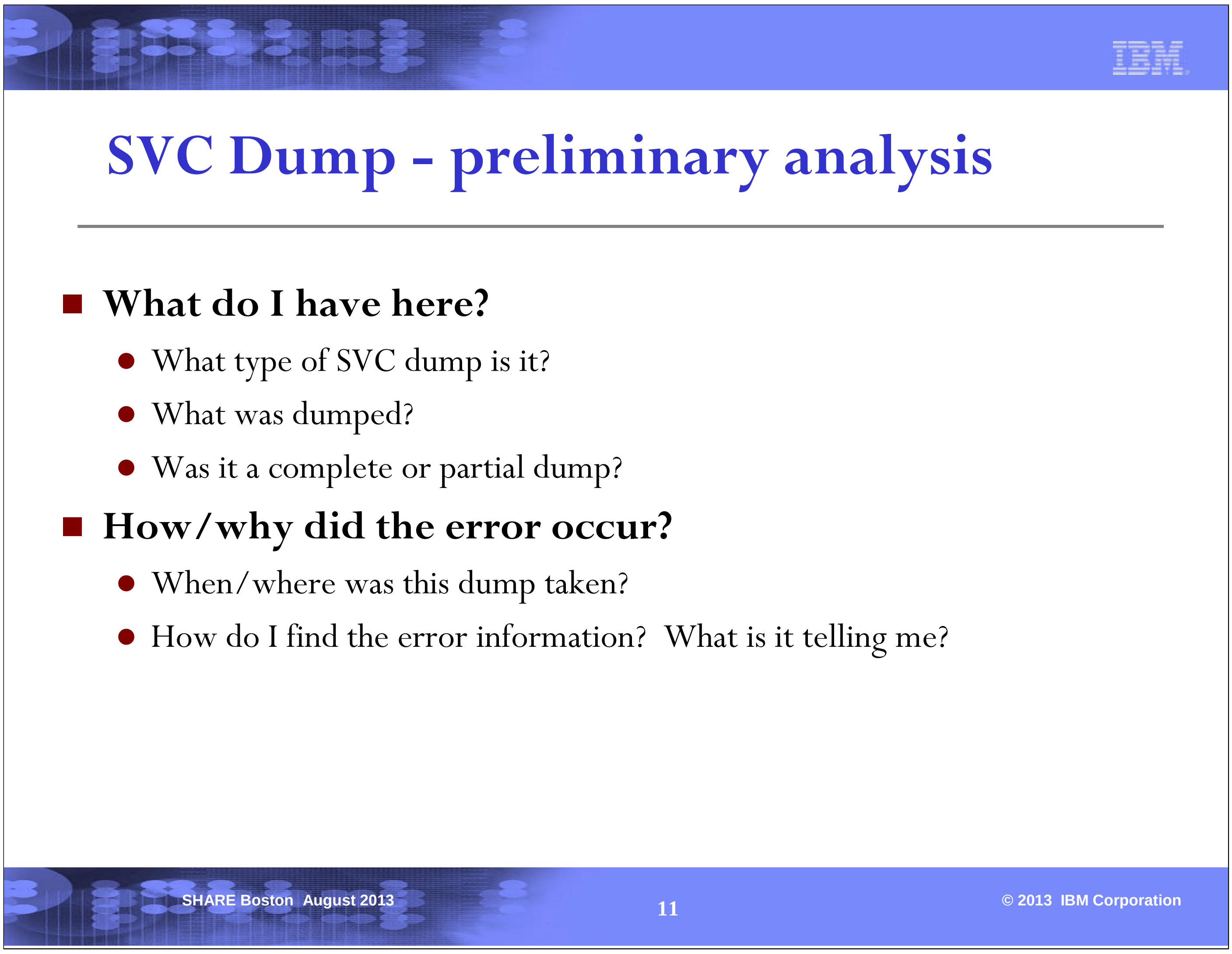
•MAXSPACE limit has been reached

•Address space being dumped is terminating or is not releasing its local lock

For MAXSPACE issues, consider whether the MAXSPACE specification needs to be increased. This is controlled through the CHNGDUMP command.

A partial dump due to an address space terminating or not releasing its local lock is environmental and cannot be prevented. In either case the partial dump should still be examined.

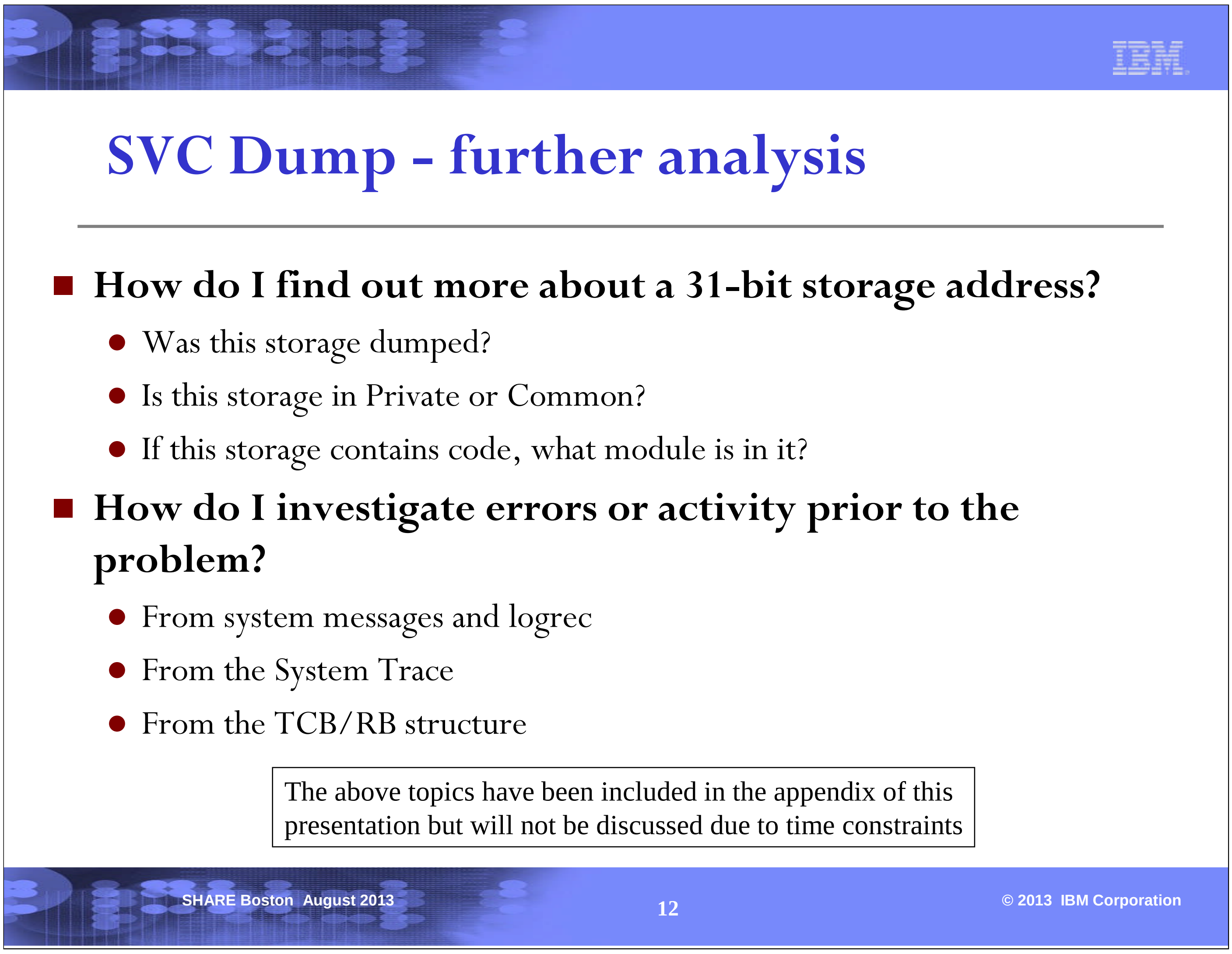
In the case of the address space terminating, there may have been earlier dumps produced. If a recreate is required, it might be possible to identify an earlier event that could trapped and dumped prior to the address space entering termination.



SVC Dump - preliminary analysis

- **What do I have here?**
  - What type of SVC dump is it?
  - What was dumped?
  - Was it a complete or partial dump?
- **How/why did the error occur?**
  - When/where was this dump taken?
  - How do I find the error information? What is it telling me?

The first thing to do when debugging an SVC dump is to find out what it contains and the reason for the dump. Then collect the error information and see if you can solve the problem. Some problems are easy, some are hard, and some are known problems!

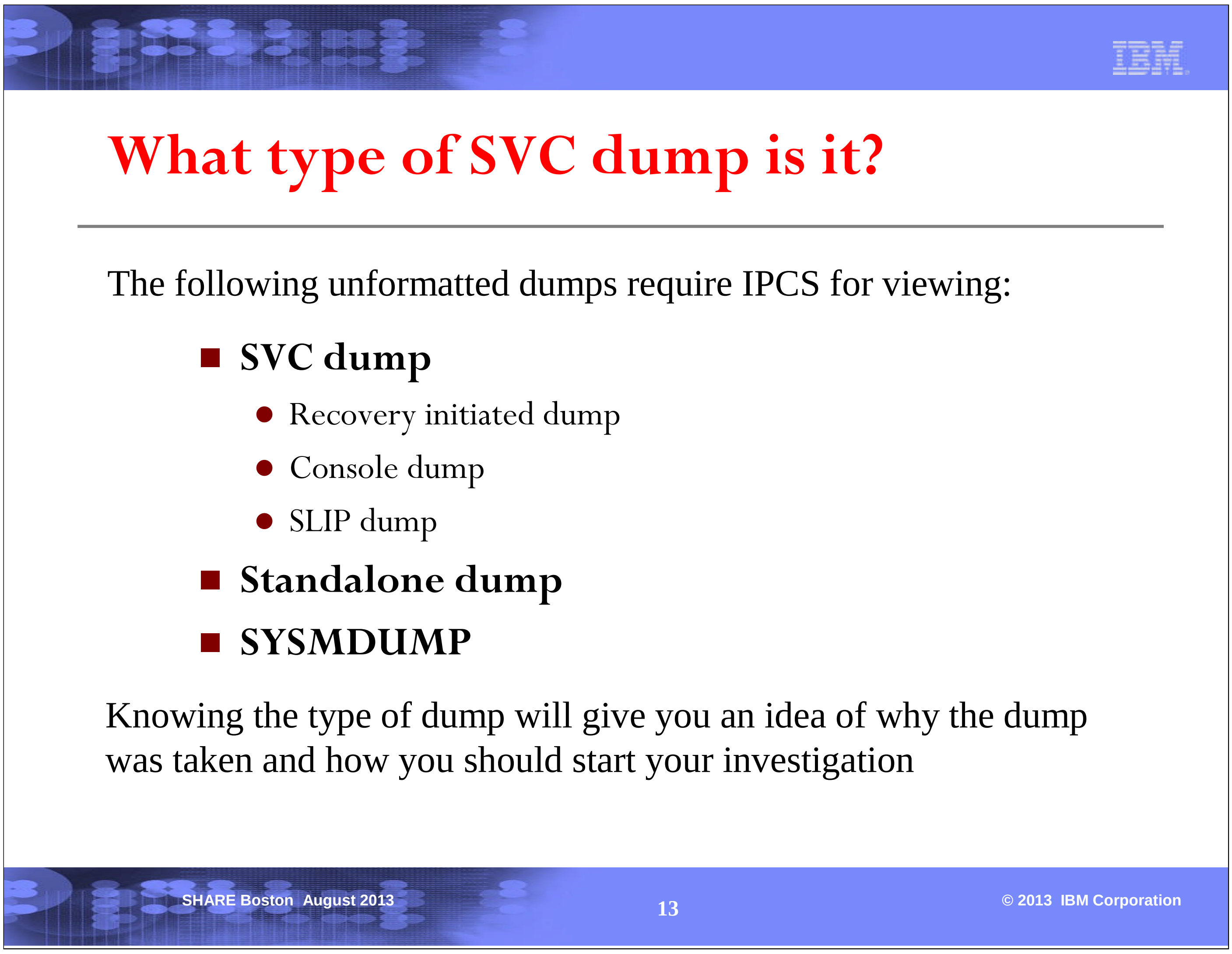


SVC Dump - further analysis

- **How do I find out more about a 31-bit storage address?**
  - Was this storage dumped?
  - Is this storage in Private or Common?
  - If this storage contains code, what module is in it?
- **How do I investigate errors or activity prior to the problem?**
  - From system messages and logrec
  - From the System Trace
  - From the TCB/RB structure

The above topics have been included in the appendix of this presentation but will not be discussed due to time constraints

This presentation includes an Appendix section containing additional diagnostic techniques of a SVC dump.



What type of SVC dump is it?

The following unformatted dumps require IPCS for viewing:

- **SVC dump**
  - Recovery initiated dump
  - Console dump
  - SLIP dump
- **Standalone dump**
- **SYSMDUMP**

Knowing the type of dump will give you an idea of why the dump was taken and how you should start your investigation

The most common type of dump is the SVC Dump. SYSMDUMP is very much like an SVC Dump but with less content. A standalone dump represents a system outage and is more complicated to work with. You may hear the term Transaction Dump (TDUMP). This is similar in content to a SYSMDUMP.

We will focus more on SVC dumps in this session.

A recovery initiated dump is taken due to an error. So one would need to find out what is the error and why it occurred.

Console dumps are taken via the DUMP command on the operator’s console. It is usually due to a problem with a job or address space. One would need to analyze the units of work in the address space. This will require reviewing the TCBs and RBs in the address space. We will touch on this near the end of the session.

SLIP dumps are taken for a purpose. One would need to find out the SLIP trap that caused the dump to be taken, then get the PSW/registers information and investigate accordingly.

IP ST SYSTEM

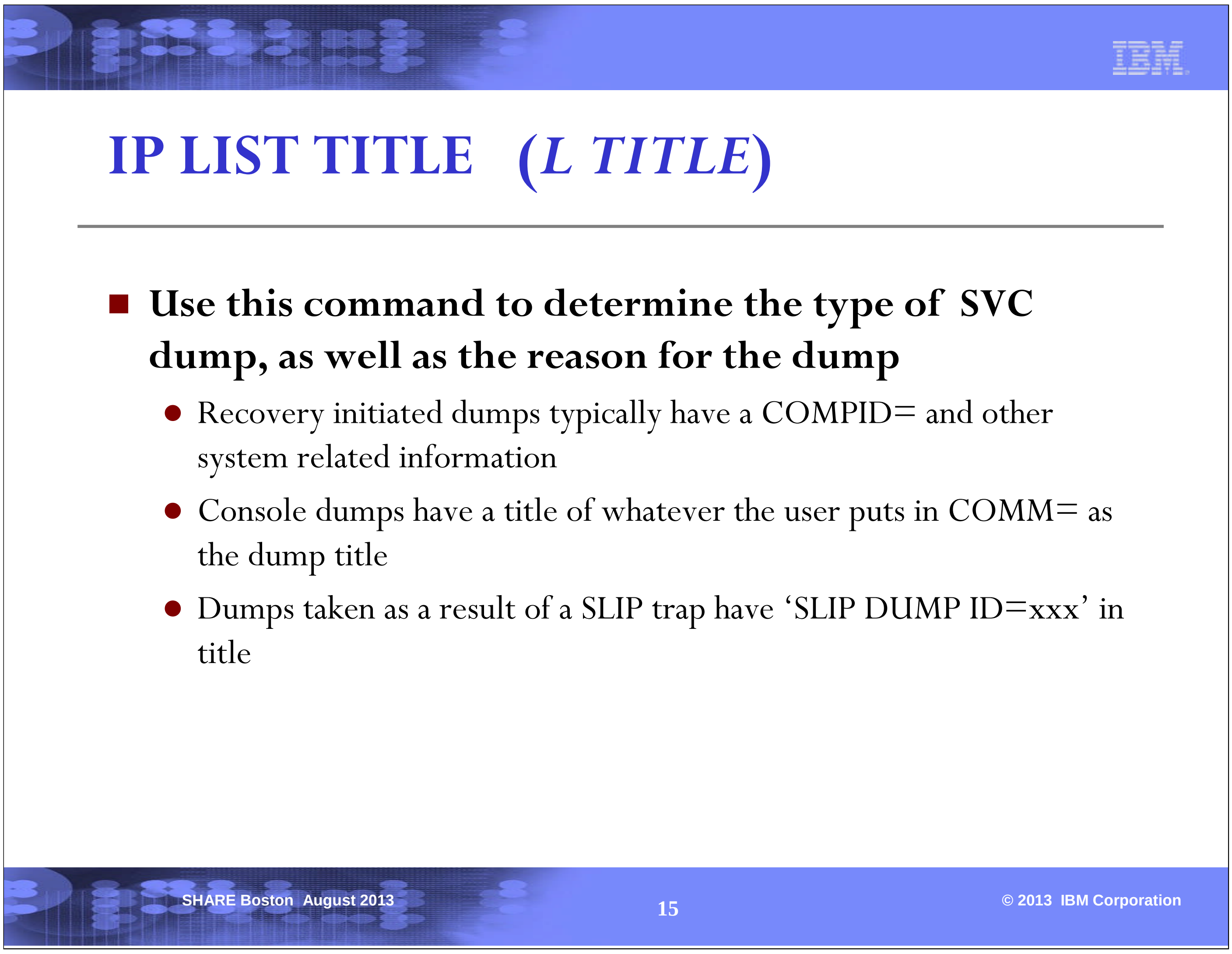
- Can be used to verify type of dump
- Program producing dump can be
  - SVCDUMP, SLIPDUMP, SYSMDUMP, SADUMP

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
   I/O configuration data:
     IODF data set name: MVSR.IODF00
     IODF configuration ID: CG21
     EDT ID: 00
  Sysplex name: TEST
  TIME OF DAY CLOCK: C1CF7E24 10F88549  01/17/2013 09:24:58.942344 local
  TIME OF DAY CLOCK: C1CF70BA D6B88549  01/17/2013 08:24:58.942344 GMT
  Program Producing Dump: SVCDUMP
  Program Requesting Dump: IEAVTSDT


  Incident token: TEST    P1N4     01/17/2013 08:24:43.288934 GMT
```

The time provided is the time that global data capture completed. This is an example of an SVC dump.

Note that SVCDUMP in the ST SYSTEM output can mean a recovery initiated SVC dump or a Console dump. To distinguish these types of dumps you need to look at the dump title (see next page).

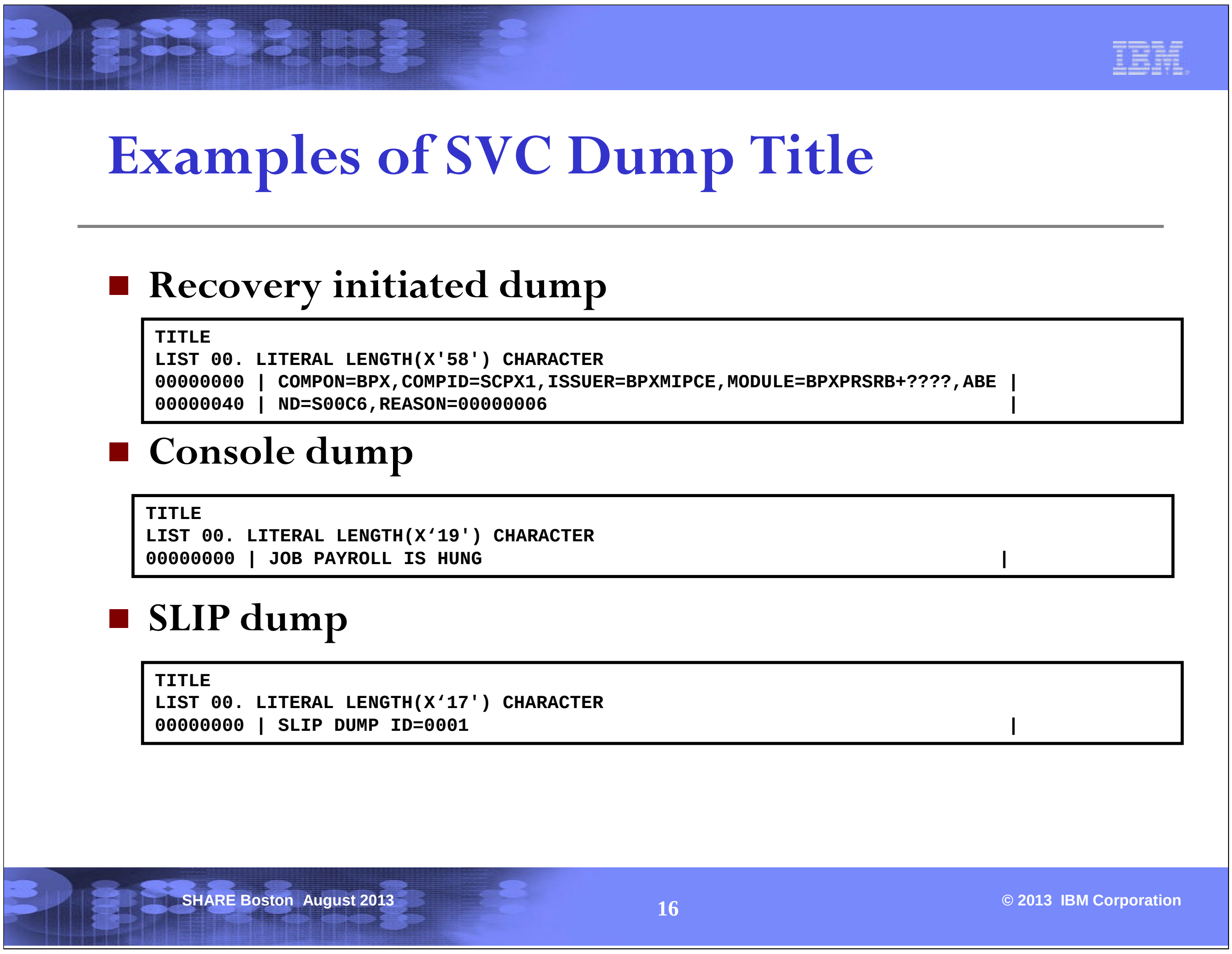
A SLIP dump is actually a type of SVC dump as well, but the ST SYSTEM command makes the distinction for us between whether it is an SVC dump generated by SLIP versus one generated by a recovery routine or operator console dump request.



IP LIST TITLE (*L TITLE*)

- **Use this command to determine the type of SVC dump, as well as the reason for the dump**
  - Recovery initiated dumps typically have a COMPID= and other system related information
  - Console dumps have a title of whatever the user puts in COMM= as the dump title
  - Dumps taken as a result of a SLIP trap have ‘SLIP DUMP ID=xxx’ in title

The dump title can provide a clue as to why the dump was taken. See the examples on the next page.



Examples of SVC Dump Title

- Recovery initiated dump

```
TITLE
LIST 00. LITERAL LENGTH(X'58') CHARACTER
00000000 | COMPON=BPX,COMPID=SCPX1,ISSUER=BPXMIPCE,MODULE=BPXPRSRB+????,ABE |
00000040 | ND=S00C6,REASON=00000006                                          |
```

- Console dump

```
TITLE
LIST 00. LITERAL LENGTH(X‘19') CHARACTER
00000000 | JOB PAYROLL IS HUNG                                               |
```

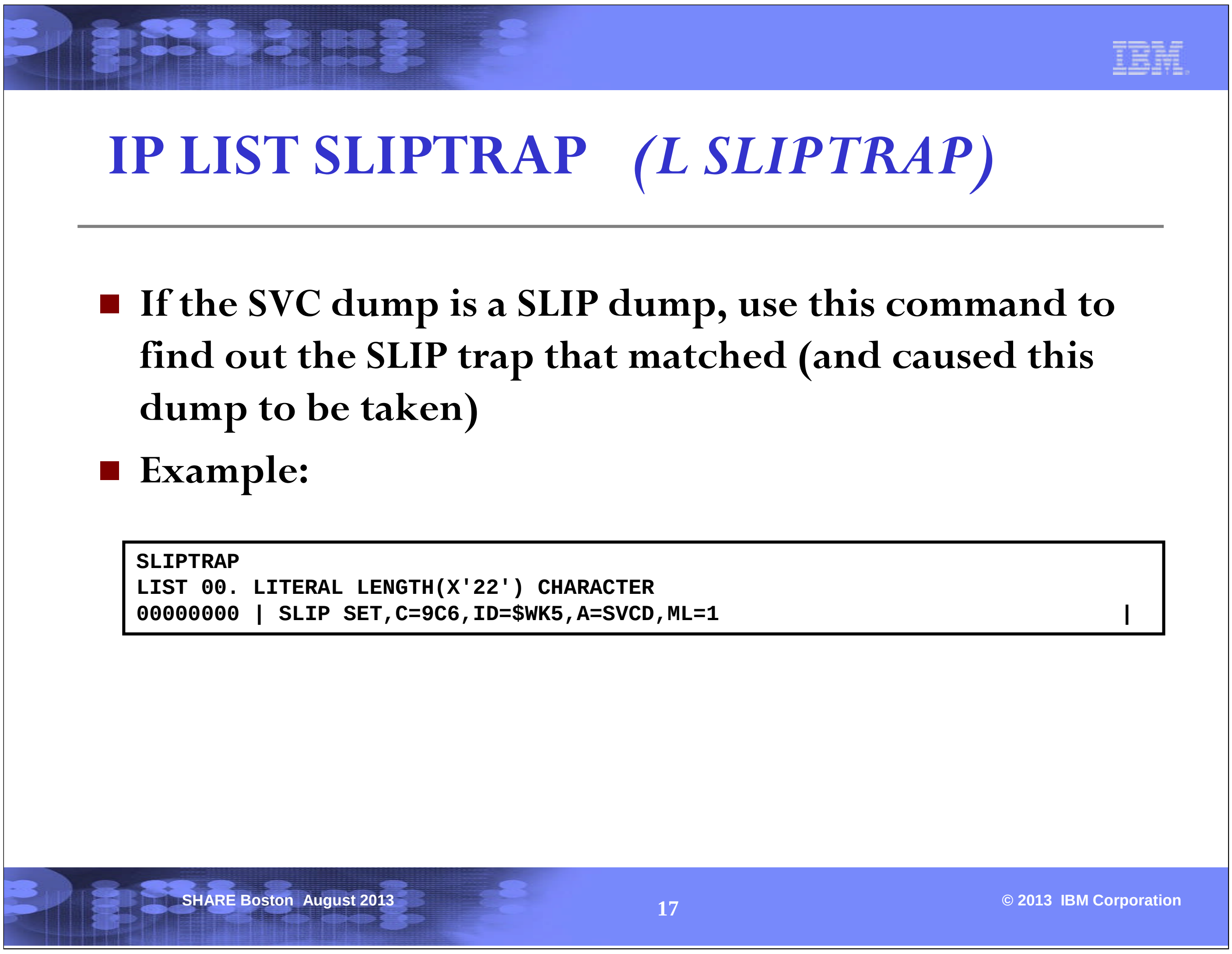
- SLIP dump

```
TITLE
LIST 00. LITERAL LENGTH(X‘17') CHARACTER
00000000 | SLIP DUMP ID=0001                                                 |
```

Recovery initiated dumps have dump titles that are pre-coded in the recovery routines. They usually contain technical information about the component or product that experienced the error.

Console dump titles are supplied by the user via the DUMP command on the console, They are usually less technical and more human-like.

SLIP dump titles are system-generated and contain the words ‘SLIP DUMP ID=‘ in it.

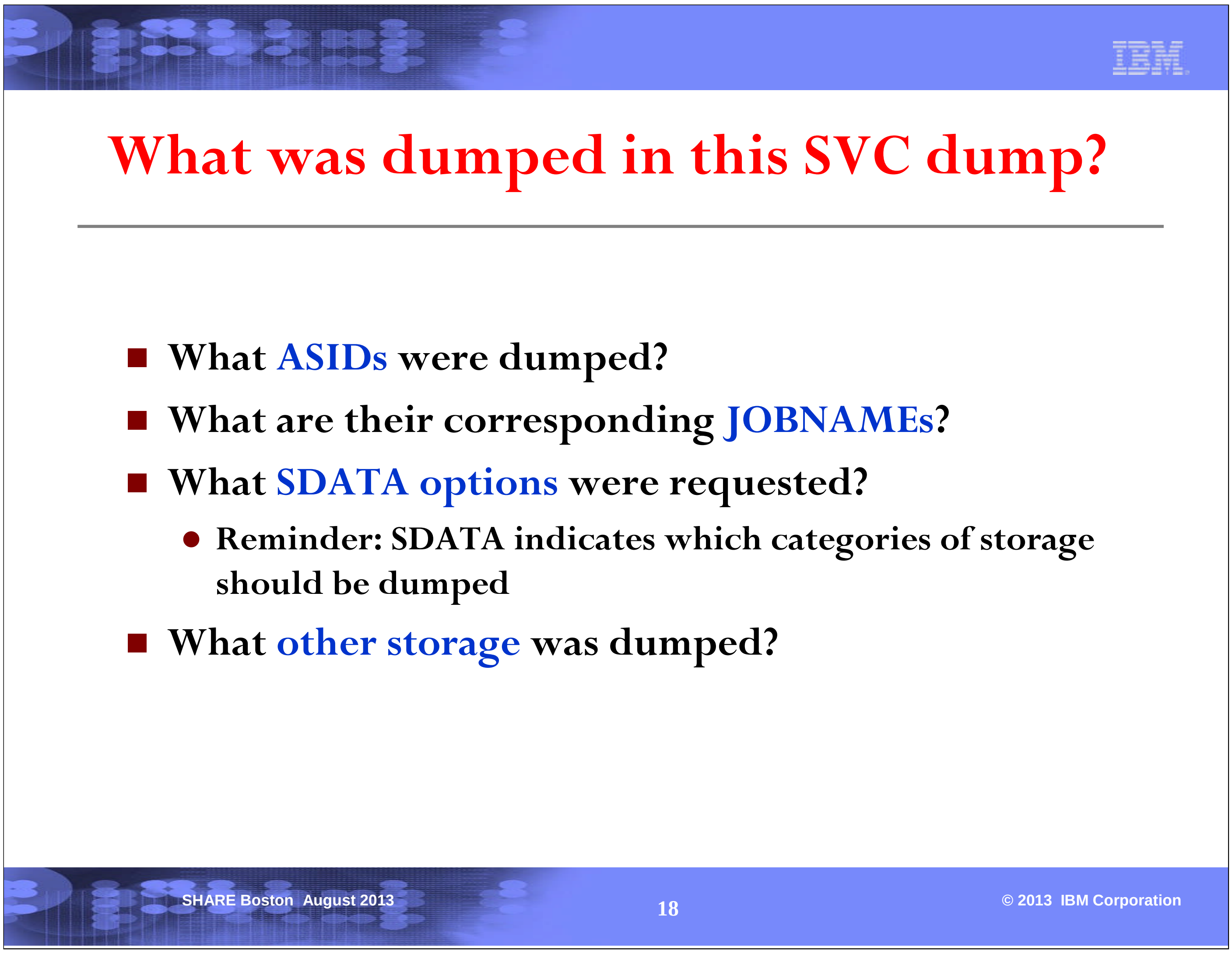


IP LIST SLIPTRAP *(L SLIPTRAP)*

- If the SVC dump is a SLIP dump, use this command to find out the SLIP trap that matched (and caused this dump to be taken)
- Example:

```
SLIPTRAP
LIST 00. LITERAL LENGTH(X'22') CHARACTER
00000000 | SLIP SET,C=9C6,ID=$WK5,A=SVCD,ML=1                                 |
```

In this example, the dump was taken as a result of a SLIP on an ABEND9C6. The ID of the SLIP is $WK5.



What was dumped in this SVC dump?

- What ASIDs were dumped?
- What are their corresponding JOBNAMEs?
- What SDATA options were requested?
  - Reminder: SDATA indicates which categories of storage should be dumped
- What other storage was dumped?

An SVC dump usually contains one or more address spaces. You will need to know the ASID numbers so that you can specify the right ones when using IPCS subcommands. (The IPCS commands do not always default to what you would expect.)

The SDATA options describe what areas of storage were requested to be in the dump.

IP CBF RTCT

- Display what ASIDs are dumped in SVC dump
- Issue 'FIND ASTB' in output

```
RTCT: 00FBFB98
   +0000  NAME..... RTCT      SAP...... FFF0BF00  SUP...... 2FD0BF00  SYD...... FF800000  SDLA..... 0000      MECB..... 809DA5A8
   +0018  FASB..... 00000000  NAS...... 00000002  EEDA..... 0220C110  SDDS..... 01DE4910  SDDC..... 0003      MTCT..... 0000
   +002C  DSV...... 00D6FDB8  SSTK..... 00000000  ADGL..... 00DD7300  ADG1..... 00DD731E  ADG2..... 00DD7324  ADG3..... 00DD732A
   +0044  ADG4..... 00DD7330  ADG5..... 00DD7F08  TABG..... 00E040D0  TABQ..... 00E040EE  TABR..... 00E04142  DSCA..... 00F8CF28
   +005C  DIND..... 02033A30  DIRS..... 024212A8  SDAT..... 02421678  SMOD..... 02232038  SCON..... 02181F80  CPID..... 021FE100
   +0074  RPAR..... 0166E4A0  BPXP..... 02538FB0  SDPL..... 02158E58  FMT...... 00000000  MLCK..... 00000001  MSRB..... 00F9D23C
   +00AC  TEST..... 00000000  SEQ#..... 11E6      SDSW..... 020CE480  RSV...... 00000000  00000000  00000000  00000000  00000000
   +00CC            00000000  00000000  00000000
00000000            SDWK..... 00D6FE88  ESEQ..... 11D7      ECPU..... 0000
   +00E4  EASD..... 0105      ETIM..... 000BC095  SAO...... FFF0BF00  SUO...... 2FD0BF00  SYO...... FF800000  SDO...... 9FE28000
   +00FE  SDNA..... 02        INDX..... 01        SDPR..... 00        BUFV..... 00000000  SDF...... 6562      ZZZ3..... 2000

       ASTB

              SDAS   SDF4   SDF5
              ----   ----   ----
          001 0105   A0     00
          002 000E   80     00
          003 0000   00     00
          004 0000   00     00
```

ASIDs dumped

Knowing which address spaces are dumped is useful for determining what address space storage you can expect to find in the dump. It may give you a clue about what address spaces are involved in the problem.

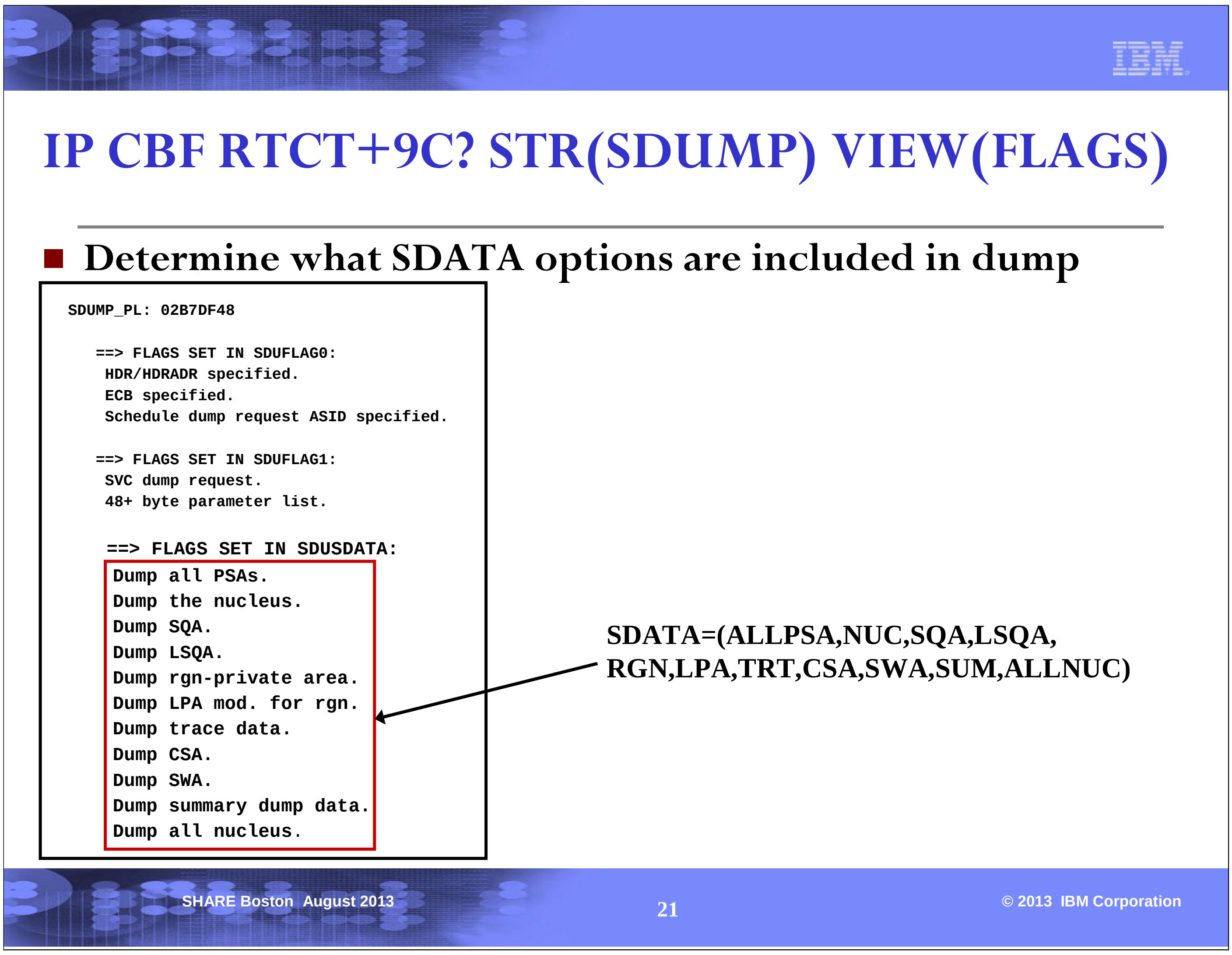
IP SELECT ALL

- ASID/JOBNAME translation

```
ASID  JOBNAME   ASCBADDR   SELECTION CRITERIA
 ---- -------- --------   ------------------
 0001 *MASTER* 00FD3900   ALL
 0002 PCAUTH   00F4DE80   ALL
 0003 RASP     00F4DD00   ALL
 0004 TRACE    00F4DB80   ALL
 0005 DUMPSRV  00F50980   ALL
 0006 XCFAS    00F50800   ALL
 0007 GRS      00F50680   ALL
 0008 SMSPDSE  00F4EF80   ALL
 0009 CONSOLE  00F4EE00   ALL
 000A WLM      00F4EC80   ALL
 000B ANTMAIN  00FC6400   ALL
 000C ANTAS000 00FC6280   ALL
 000D DEVMAN   00FC6100   ALL
 000E OMVS     00FA2800   ALL
```

- Or IP SELECT ASID(x'nn') / IP SELECT JOB(jobname)

This report associates an ASID with a JOBNAME. From the CBF RTCT example on the previous page, we can see that asid(x’E’) corresponds to a jobname of OMVS.



IP CBF RTCT+9C? STR(SDUMP) VIEW(FLAGS)

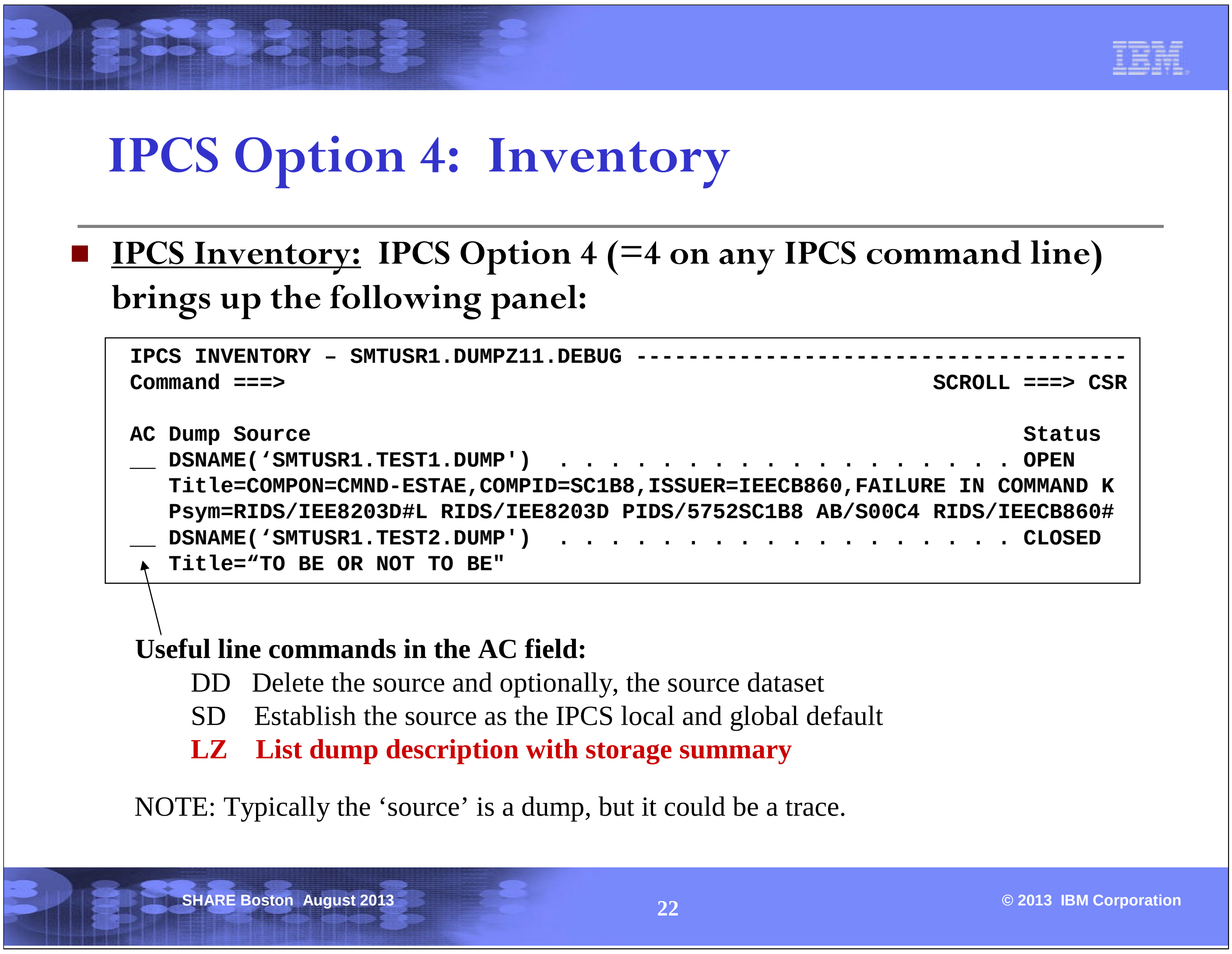
- Determine what SDATA options are included in dump

```
SDUMP_PL: 02B7DF48

   ==> FLAGS SET IN SDUFLAG0:
    HDR/HDRADR specified.
    ECB specified.
    Schedule dump request ASID specified.

   ==> FLAGS SET IN SDUFLAG1:
    SVC dump request.
    48+ byte parameter list.

    ==> FLAGS SET IN SDUSDATA:
     Dump all PSAs.
     Dump the nucleus.
     Dump SQA.
     Dump LSQA.
     Dump rgn-private area.
     Dump LPA mod. for rgn.
     Dump trace data.
     Dump CSA.
     Dump SWA.
     Dump summary dump data.
     Dump all nucleus.
```

SDATA=(ALLPSA,NUC,SQA,LSQA,
RGN,LPA,TRT,CSA,SWA,SUM,ALLNUC)

SHARE Boston August 2013 21 © 2013 IBM Corporation

Knowing the dump options that were requested can help you determine why the storage or report that you’re browsing is not available. This is because the option that would have included the relevant storage was not requested. For example, if you’re trying to browse private storage of an address space, and the storage is not available, it may be due to RGN not being specified in SDATA.



IPCS Option 4: Inventory

- **IPCS Inventory:** IPCS Option 4 (=4 on any IPCS command line) brings up the following panel:

```
IPCS INVENTORY – SMTUSR1.DUMPZ11.DEBUG ------------------------------------
Command ===>                                                 SCROLL ===> CSR

AC Dump Source                                                      Status
__ DSNAME(‘SMTUSR1.TEST1.DUMP') . . . . . . . . . . . . . . . . . . OPEN
   Title=COMPON=CMND-ESTAE,COMPID=SC1B8,ISSUER=IEECB860,FAILURE IN COMMAND K
   Psym=RIDS/IEE8203D#L RIDS/IEE8203D PIDS/5752SC1B8 AB/S00C4 RIDS/IEECB860#
__ DSNAME(‘SMTUSR1.TEST2.DUMP') . . . . . . . . . . . . . . . . . . CLOSED
   Title=“TO BE OR NOT TO BE"
```

Useful line commands in the AC field:

DD	Delete the source and optionally, the source dataset
SD	Establish the source as the IPCS local and global default
LZ	**List dump description with storage summary**

NOTE: Typically the ‘source’ is a dump, but it could be a trace.

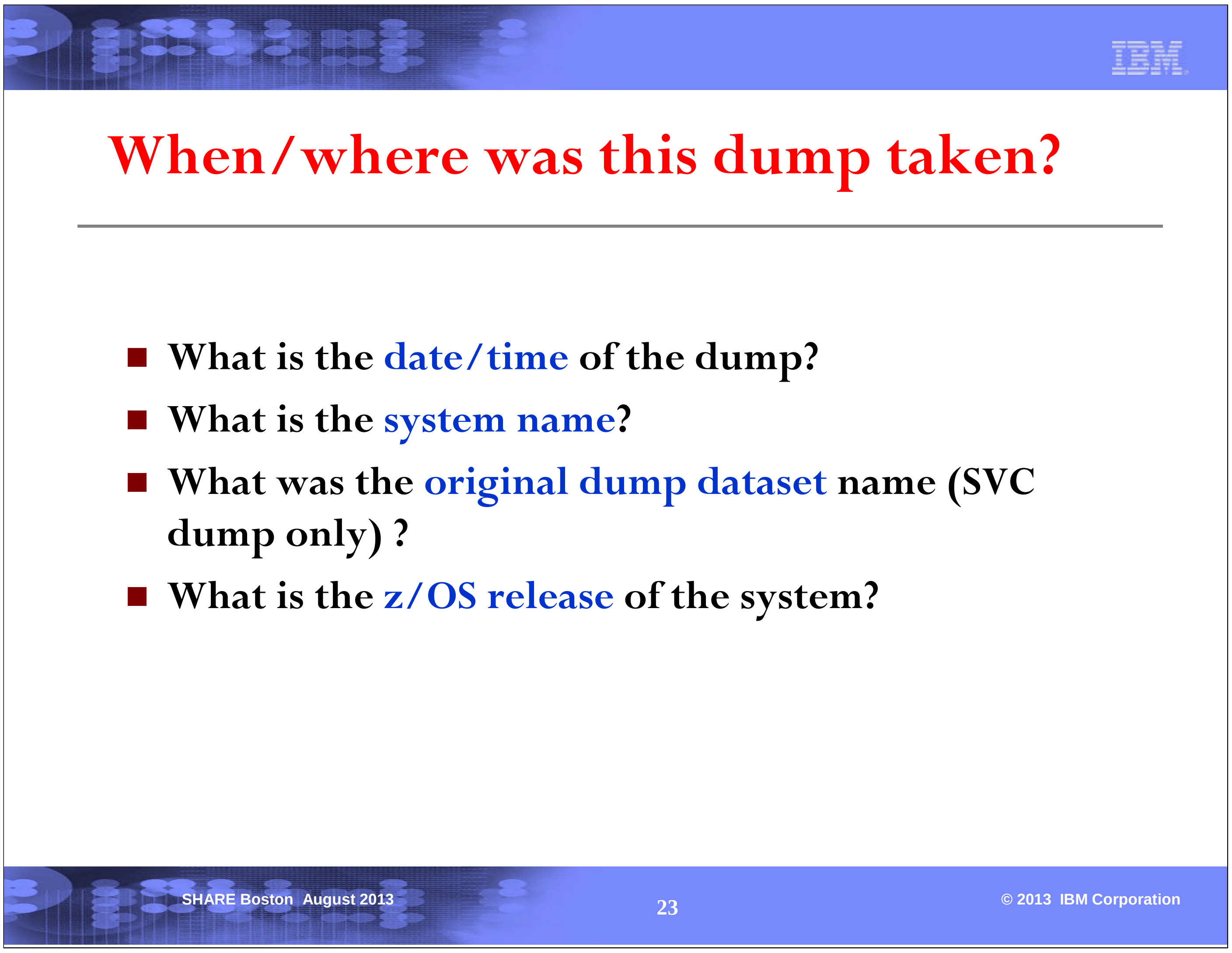
The IPCS inventory panel allows the user to view all the dumps in his IPCS dump directory. The IPCS inventory panel shows all the dumps the user has initialized under IPCS and is currently working with.

One can enter a line command in the column AC next to a DSNAME, and then hit <enter>.

SD selects a dump or trace to be used as the current source for IPCS.

DD deletes information of a dump or trace from the IPCS directory. This is needed when the user does not need to review the dump or trace anymore. It will disappear from the IPCS inventory.

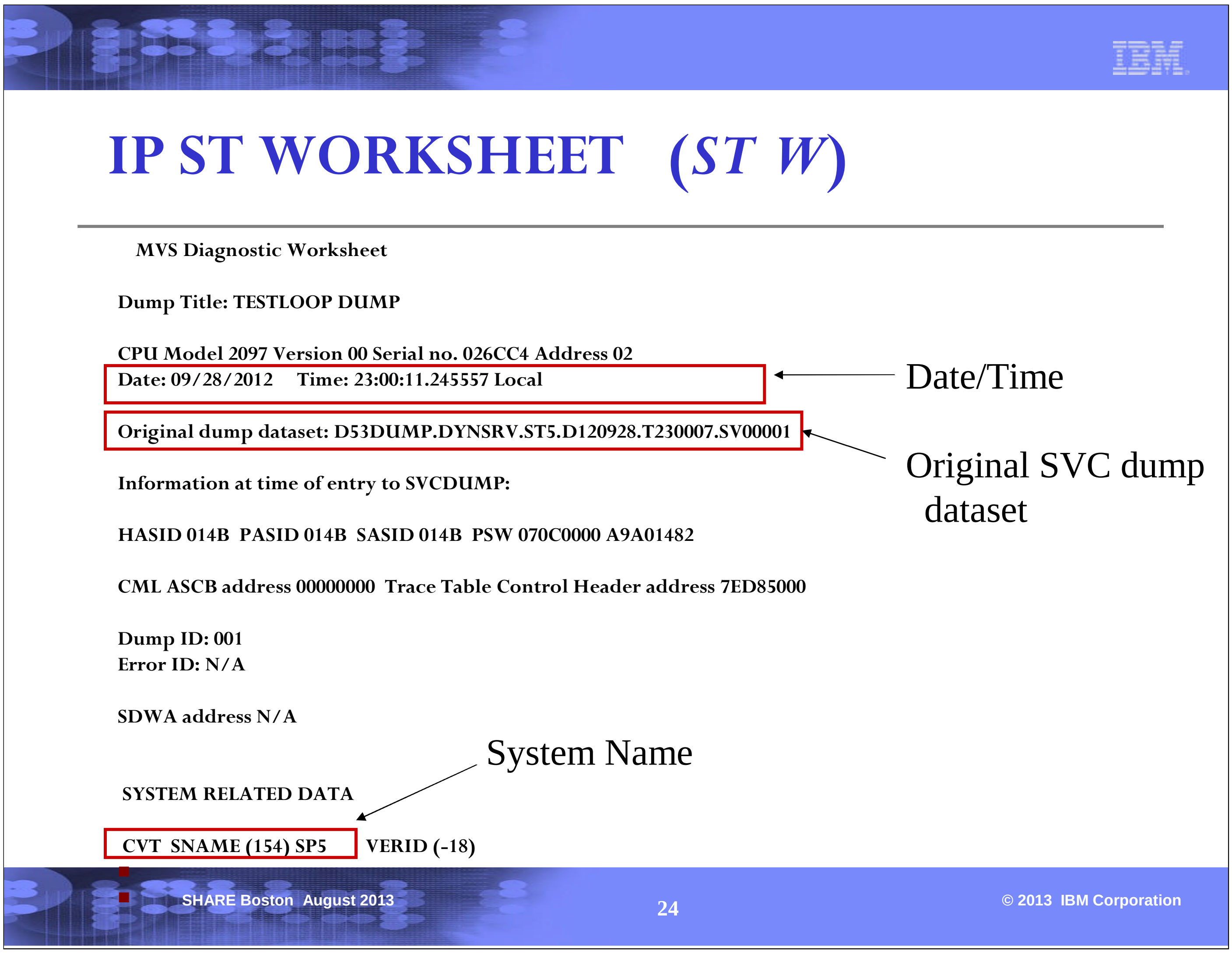
LZ gives detailed and rather cryptic information about the address spaces, dataspaces and storage included in a dump.



When/where was this dump taken?

- What is the date/time of the dump?
- What is the system name?
- What was the original dump dataset name (SVC dump only) ?
- What is the z/OS release of the system?

It is important to find out when and where the dump was taken. You may need to correlate this dump with other events occurring at the same time on this system or other systems.



IP ST WORKSHEET (*ST W*)

```
   MVS Diagnostic Worksheet

Dump Title: TESTLOOP DUMP

CPU Model 2097 Version 00 Serial no. 026CC4 Address 02
Date: 09/28/2012     Time: 23:00:11.245557 Local

Original dump dataset: D53DUMP.DYNSRV.ST5.D120928.T230007.SV00001

Information at time of entry to SVCDUMP:

HASID 014B  PASID 014B  SASID 014B  PSW 070C0000 A9A01482

CML ASCB address 00000000  Trace Table Control Header address 7ED85000

Dump ID: 001
Error ID: N/A

SDWA address N/A


 SYSTEM RELATED DATA

 CVT  SNAME (154) SP5     VERID (-18)
```

This is an example of a SVC dump taken on system SP5 on 9/28/12.   The time is actually the time of the end of the global data capture phase of SVC dump processing.

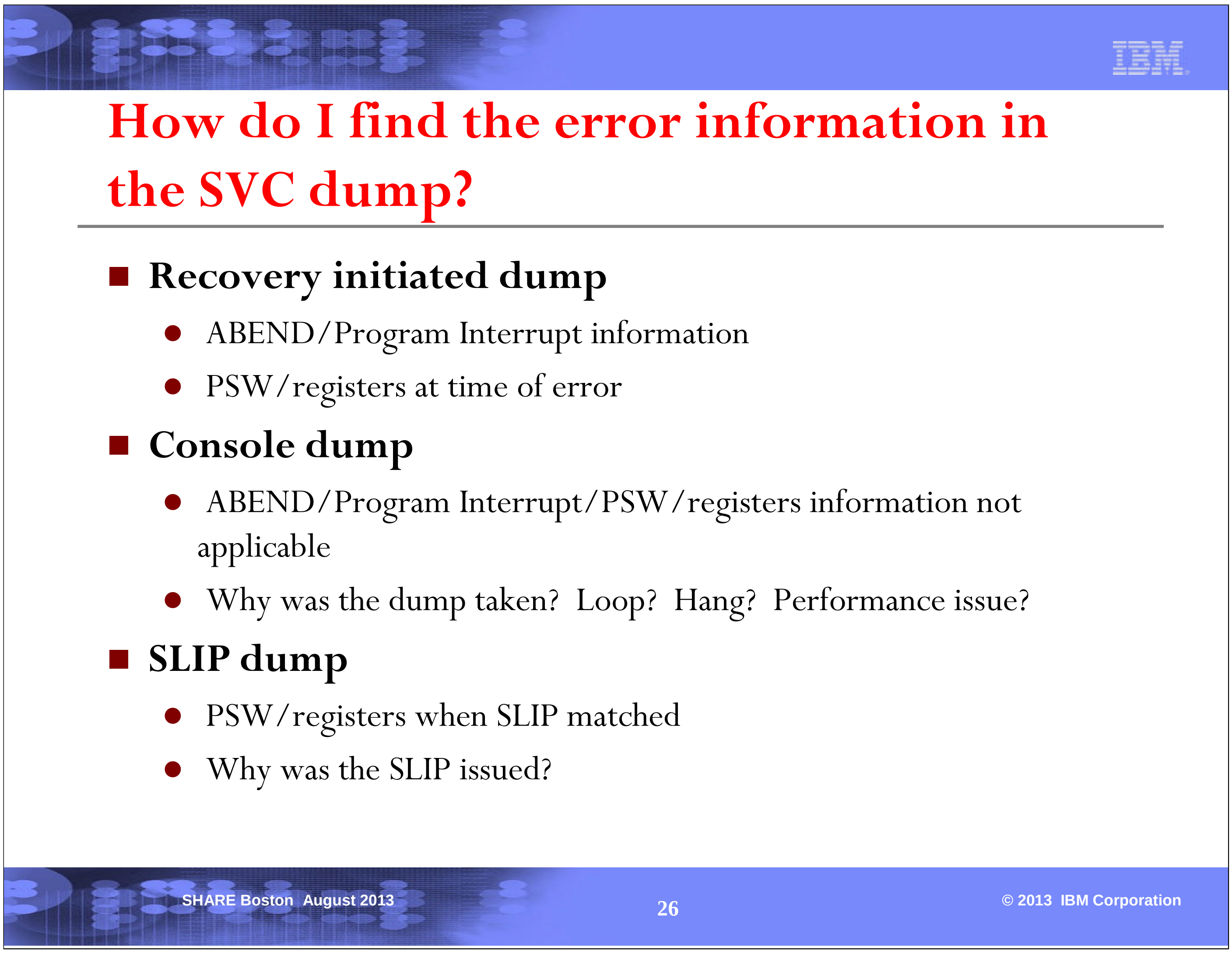
IP CBF CVT

- Can be used to verify system release level on which dump was taken
- Example:

```
CVT: 00FDEAC0
   -0028  PRODN.... SP7.1.3   PRODI.... HBB7780   VERID....
```

- In the above example, dump was taken on a z/OS R13 system

This is important because you should use the same level of IPCS as the z/OS release of the system on which the dump was taken. IPCS will put out a message if you use a level of IPCS that is different than the level of the z/OS system on which the dump was taken.



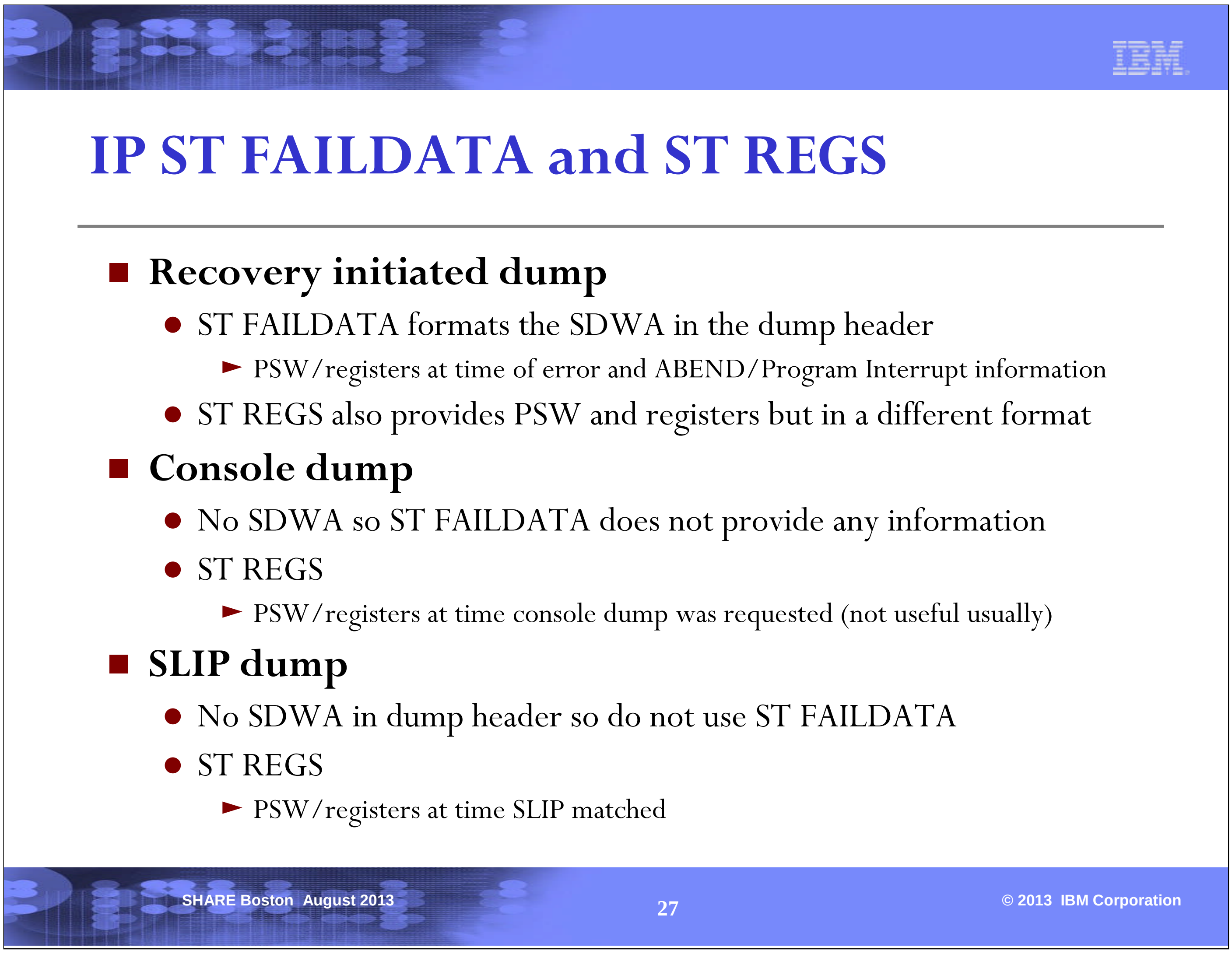
How do I find the error information in the SVC dump?

- **Recovery initiated dump**
  - ABEND/Program Interrupt information
  - PSW/registers at time of error
- **Console dump**
  - ABEND/Program Interrupt/PSW/registers information not applicable
  - Why was the dump taken? Loop? Hang? Performance issue?
- **SLIP dump**
  - PSW/registers when SLIP matched
  - Why was the SLIP issued?

There is always a set of PSW/register information crucial for debugging a recovery-initiated dump or SLIP dump.

For a console dump, one needs to investigate the state of an address space, and there is no PSW/register set of interest initially.

For a standalone dump, one needs to investigate the state of the whole system. The PSW/registers of each CPU is a good place to start, but there are many other pieces of information to be reviewed. Details of how to investigate a standalone dump will not be covered in this session.



IP ST FAILDATA and ST REGS

- **Recovery initiated dump**
  - ST FAILDATA formats the SDWA in the dump header
    - PSW/registers at time of error and ABEND/Program Interrupt information
  - ST REGS also provides PSW and registers but in a different format
- **Console dump**
  - No SDWA so ST FAILDATA does not provide any information
  - ST REGS
    - PSW/registers at time console dump was requested (not useful usually)
- **SLIP dump**
  - No SDWA in dump header so do not use ST FAILDATA
  - ST REGS
    - PSW/registers at time SLIP matched

SHARE Boston August 2013 27 © 2013 IBM Corporation

An SVC dump may be captured as the result of:

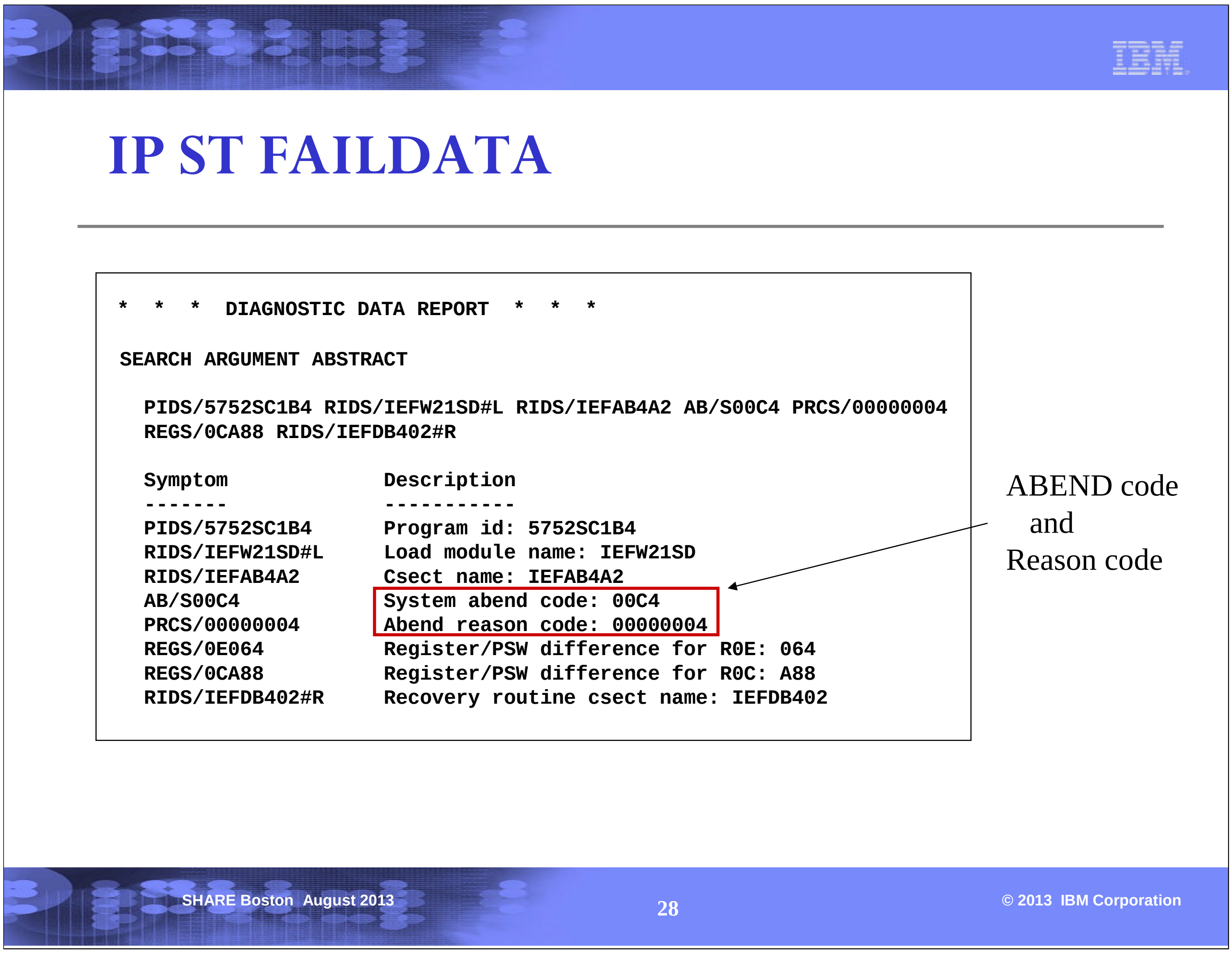
•A program recognizing that an error has occurred and invoking recovery

•As a result of a SLIP trap

•Because the operator/system programmer requests a dump.

The most basic and crucial piece of debugging information is typically the PSW and register information at the time of error or failure.

The availability of this information will depend on how the dump was taken. ST FAILDATA typically gives the most information about a problem, but is only available if a populated SDWA (System Diagnostic Work Area) is available. This will only be the case if a recovery routine requests that the dump be captured. In a console dump, the registers found in ST REGS show what was happening at the time the dump was requested, which means the program running at the time is not likely the one of interest, and the PSW and registers are less useful. In a SLIP generated dump, there is no SDWA, but the PSW/registers may be of value since they are captured at the time the SLIP trap springs.

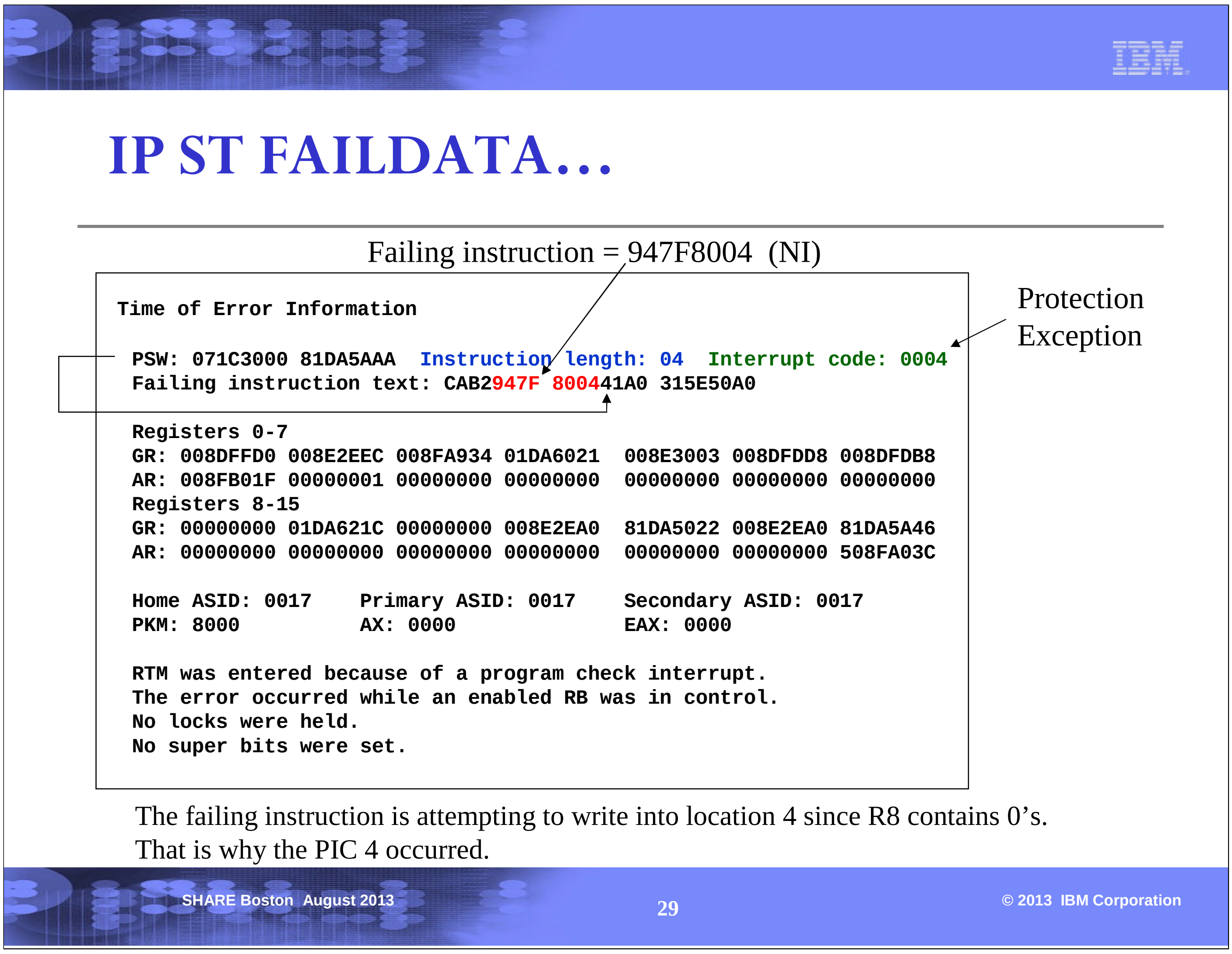


IP ST FAILDATA

```
*   *   *  DIAGNOSTIC DATA REPORT  *   *   *

SEARCH ARGUMENT ABSTRACT

  PIDS/5752SC1B4 RIDS/IEFW21SD#L RIDS/IEFAB4A2 AB/S00C4 PRCS/00000004
  REGS/0CA88 RIDS/IEFDB402#R

  Symptom              Description
  -------              -----------
  PIDS/5752SC1B4       Program id: 5752SC1B4
  RIDS/IEFW21SD#L      Load module name: IEFW21SD
  RIDS/IEFAB4A2        Csect name: IEFAB4A2
  AB/S00C4             System abend code: 00C4
  PRCS/00000004        Abend reason code: 00000004
  REGS/0E064           Register/PSW difference for R0E: 064
  REGS/0CA88           Register/PSW difference for R0C: A88
  RIDS/IEFDB402#R      Recovery routine csect name: IEFDB402
```

ABEND code
and
Reason code

The first page of the ST FAILDATA output contains information about the program involved with the error, as well as the ABEND code and reason code.



IP ST FAILDATA…

Failing instruction = 947F8004 (NI)

```
Time of Error Information

  PSW: 071C3000 81DA5AAA  Instruction length: 04  Interrupt code: 0004
  Failing instruction text: CAB2947F 800441A0 315E50A0

  Registers 0-7
  GR: 008DFFD0 008E2EEC 008FA934 01DA6021  008E3003 008DFDD8 008DFDB8
  AR: 008FB01F 00000001 00000000 00000000  00000000 00000000 00000000
  Registers 8-15
  GR: 00000000 01DA621C 00000000 008E2EA0  81DA5022 008E2EA0 81DA5A46
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 508FA03C

  Home ASID: 0017    Primary ASID: 0017    Secondary ASID: 0017
  PKM: 8000          AX: 0000              EAX: 0000

  RTM was entered because of a program check interrupt.
  The error occurred while an enabled RB was in control.
  No locks were held.
  No super bits were set.
```

Protection Exception

The failing instruction is attempting to write into location 4 since R8 contains 0's. That is why the PIC 4 occurred.

In the case of a program check, find out the PIC (Program Interrupt Code) from the Interrupt code. In this example, it is a PIC4 (Protection Exception).

For PIC 10,11,38,39,3A,3B the PSW at time of error points at the failing instruction.

For PIC 4, the PSW points after the failing instruction.

When reviewing the Failing instruction text, start in the middle (6 bytes into text), and backup the number of bytes (if necessary) specified by the Instruction length. For a PIC 4, we back up by the Instruction length (4 bytes in this case).

In the machine instruction 947F8004, the base register position holds an 8. This means that this instruction, an aNd Immediate, is attempting to alter storage pointed to by register 8. Register 8 contains zero, so this instruction is trying to update low core location zero. Thankfully, there is extra protection on this critical area of system storage, so an ABEND0C4 PIC4 results, preventing an overlay. Note that in this example, the PSW execution key is 1, so even without special low core protection, the NI operation would have failed, since the low core storage is key0, which does not match the execution key.

IP ST REGS

```
CPU STATUS:

PSW=071C3000  81DA5AAA  (RUNNING IN PRIMARY, KEY 1,  AMODE 31, DAT ON)
    DISABLED FOR PER
 ASID(X'0017') 01DA5AAA. IEFW21SD+0AAA IN EXTENDED PLPA
 ASCB23 at F85380, JOB(JERRY), for the home ASID
 ASXB23 at 8FDF00 for the home ASID. No block is dispatched
 HOME ASID: 0017 PRIMARY ASID: 0017 SECONDARY ASID: 0017

 GPR VALUES
     0-3  008DFFD0  008E2EEC  008FA934  01DA6021
     4-7  008E3003  008DFDD8  008DFDB8  008E02C8
     8-11 00000000  01DA621C  00000000  008E2EA0
    12-15 81DA5022  008E2EA0  81DA5A46  81DA62E0

 ACCESS REGISTER VALUES
     0-3  008FB01F  00000001  00000000  00000000
     4-7  00000000  00000000  00000000  00000000
     8-11 00000000  01DA621C  00000000  008E2EA0
    12-15 81DA5022  008E2EA0  81DA5A46  81DA62E0
```

Additional information on PSW

This report is not dependent on an SDWA (as is ST FAILDATA), and is useful for both SLIP dumps and dumps generated by a recovery routine. It provides more information about the PSW, but the Failing Instruction Text, Instruction Length, and Interrupt Code are not provided (as with ST FAILDATA).

It also provides contents of Access registers and Control registers (not shown above).

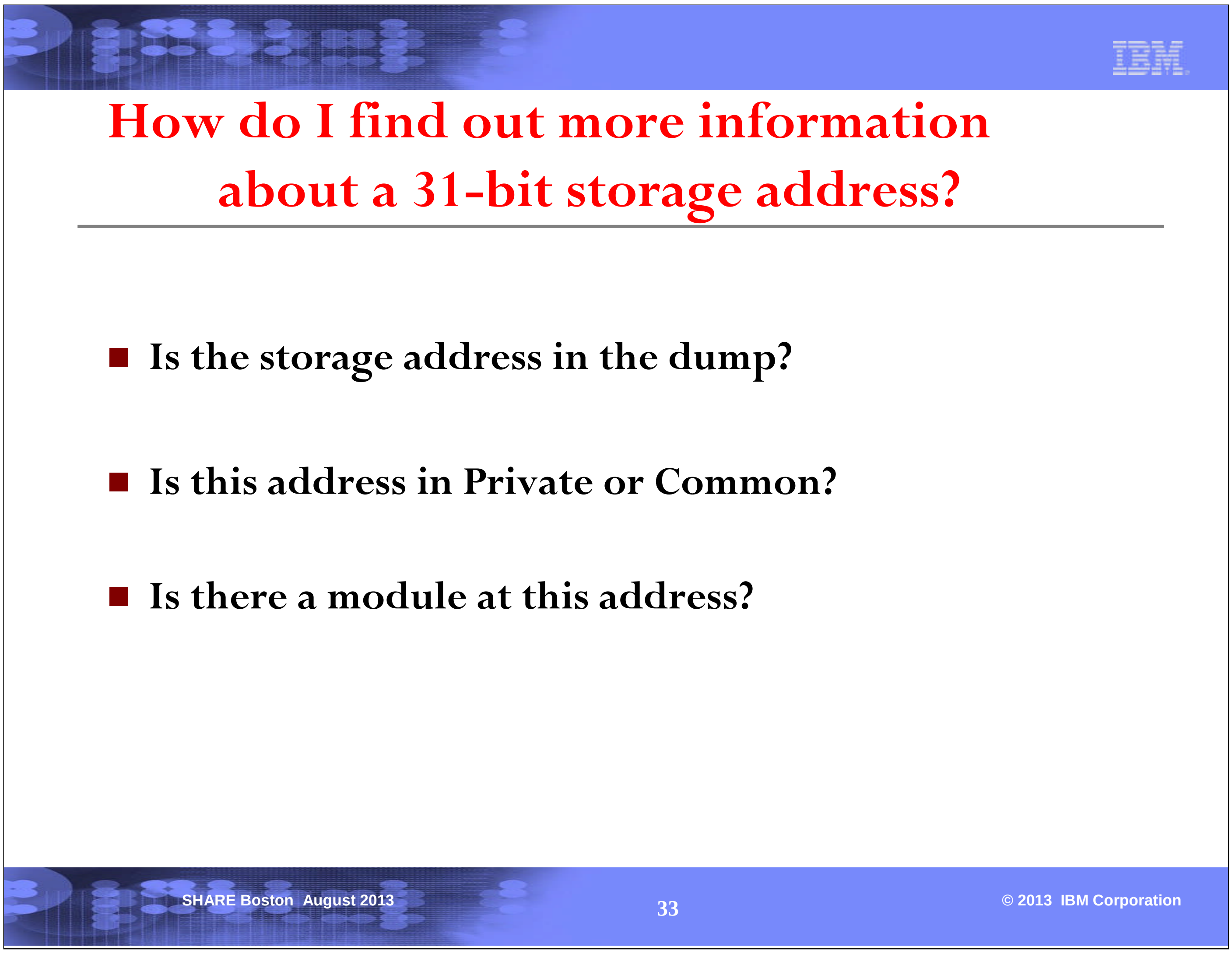
Reference Information

- **Manuals**
  - z/OS MVS IPCS Commands
  - z/OS MVS IPCS Customization
  - z/OS MVS IPCS User's Guide
  - z/OS MVS Diagnosis: Reference
  - z/OS MVS Diagnosis: Tools and Service Aids
  - z/OS MVS System Codes
  - z/OS MVS Data Areas

Appendix - further SVC dump analysis

The following pages contain additional material that will not be discussed in this session

End



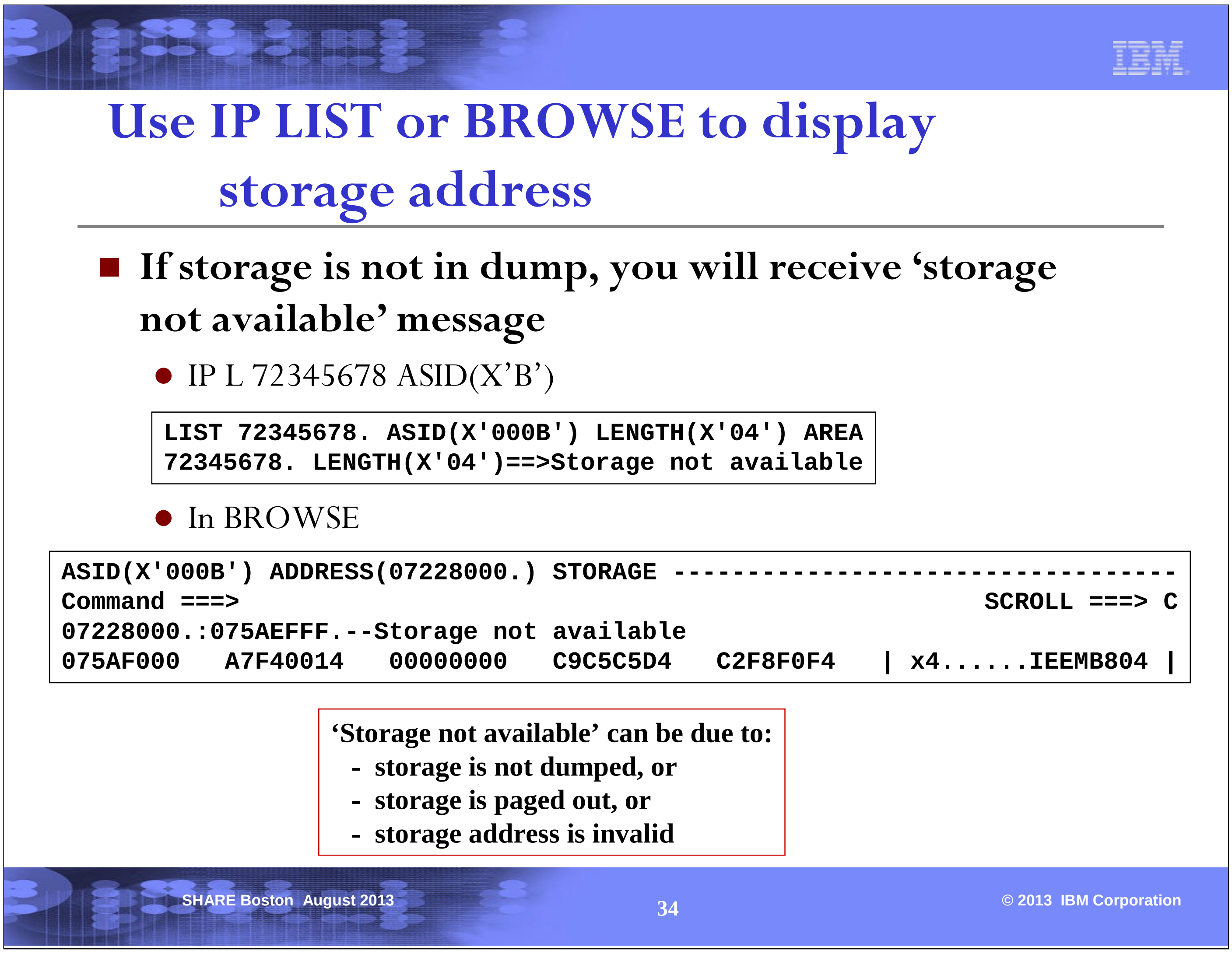
How do I find out more information about a 31-bit storage address?

- Is the storage address in the dump?
- Is this address in Private or Common?
- Is there a module at this address?

SHARE Boston August 2013 33 © 2013 IBM Corporation

When you are debugging a problem with a set of PSW and registers, it is important that you know where the PSW and registers point to. This section will answer some of the questions you may have on an address.

If the address is in common storage, an ASID parameter is not needed when displaying the storage.



Use IP LIST or BROWSE to display storage address

- If storage is not in dump, you will receive ‘storage not available’ message
  - IP L 72345678 ASID(X’B’)

```
LIST 72345678. ASID(X'000B') LENGTH(X'04') AREA
72345678. LENGTH(X'04')==>Storage not available
```

  - In BROWSE

```
ASID(X'000B') ADDRESS(07228000.) STORAGE ---------------------------------
Command ===>                                                  SCROLL ===> C
07228000.:075AEFFF.--Storage not available
075AF000    A7F40014    00000000    C9C5C5D4    C2F8F0F4    | x4......IEEMB804 |
```

‘Storage not available’ can be due to:
- **storage is not dumped, or**
- **storage is paged out, or**
- **storage address is invalid**

There are several reasons for ‘storage not available’, and it also depends on the kind of dump.

Since everything should be dumped in a standalone dump, ‘storage not available’ most likely means that the storage address is invalid.

In an SVC dump or SYSMDUMP, some common areas (for example, LPA) are usually not dumped. So it is important to know which area the storage address belongs to (see next page).

IP VERBX VSMDATA ‘NOASIDS SUMM’

- Scroll to the bottom and then page back to find the Global Storage Map

```
GLOBAL STORAGE MAP
                                         80000000  <- Top Of Storage
   ___________________________________
  :                                   :
  :   Extended Private Storage        :
  :___________________________________:2A800000  <- Ext. CSA Upper Bound
  |                                   |
  |   Extended CSA                    |
  |___________________________________| B385000  <- Ext. CSA Lower Bound
  |                                   |
  |   Extended PLPA/FLPA/MLPA         |
  |___________________________________| 65EB000  <- Ext. SQA Upper Bound
  |                                   |
  |   Extended SQA                    |
```

‘

Only the top part of the storage map is shown here

VERBX VSMDATA with a parameter of NOASIDS provides a global storage map that shows the boundaries of different area of storage for every address space. This will allow you to figure out whether the address at hand is in private or common. To see the details of the private storage of an address space see next page.

IBM

IP VERBX VSMDATA ‘ASID(nn) SUMM’ (nn in decimal)

- Scroll to the bottom and then page back to find the Local Storage Map

```
 LOCAL STORAGE MAP
 ___________________________
|                           |80000000  <- Top of Ext. Private
| Extended                  |
| LSQA/SWA/229/230          |80000000  <- Max Ext. User Region Address
|___________________________|7EB73000  <- ELSQA Bottom
|                           |
| (Free Extended Storage)   |
|___________________________|2A9C9000  <- Ext. User Region Top
|                           |
| Extended User Region      |
‘|___________________________|2A800000  <- Ext. User Region Start
:                           :
```

Only the top part of the storage map is shown here

To see the details of the layout of private storage in a particular address space, use VERBX VSMDATA with an ASID parameter. This is useful after you have determined that the address at hand is in the private region of an address space.

IP WHERE (*W*)

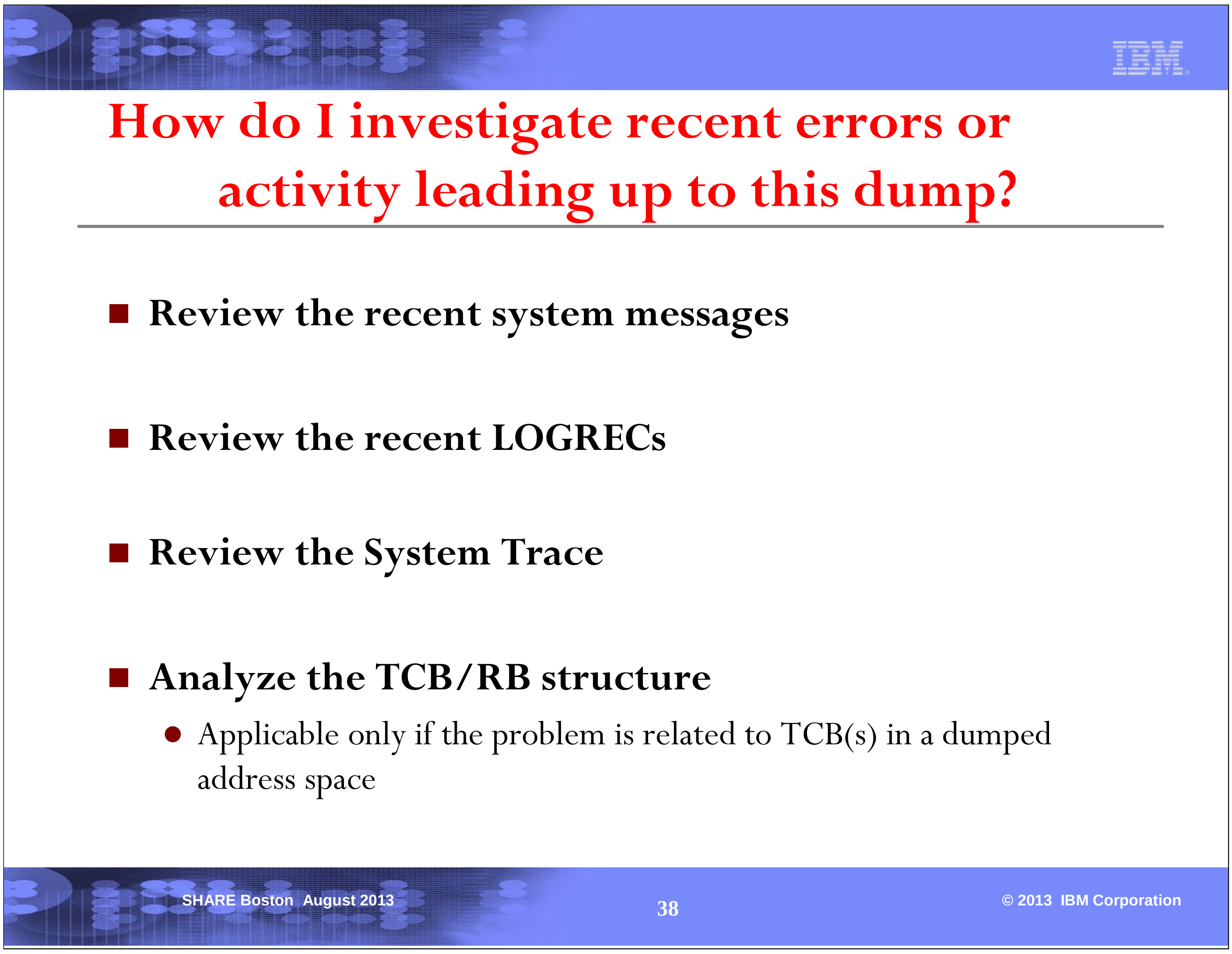
- Used to identify an area in the dump (if possible)
  - IP W 75AF0D0

```
Command ===>
 ****************************** TOP OF DATA ***************
      ASID(X'000B') 075AF0D0. IGC0003F+D0 IN EXTENDED PLPA
 ****************************** END OF DATA ***************
```

► ASID(x'nn') should be used if storage address is in private

- If WHERE cannot identify the area, BROWSE the storage and scan backwards for module or control block identifiers

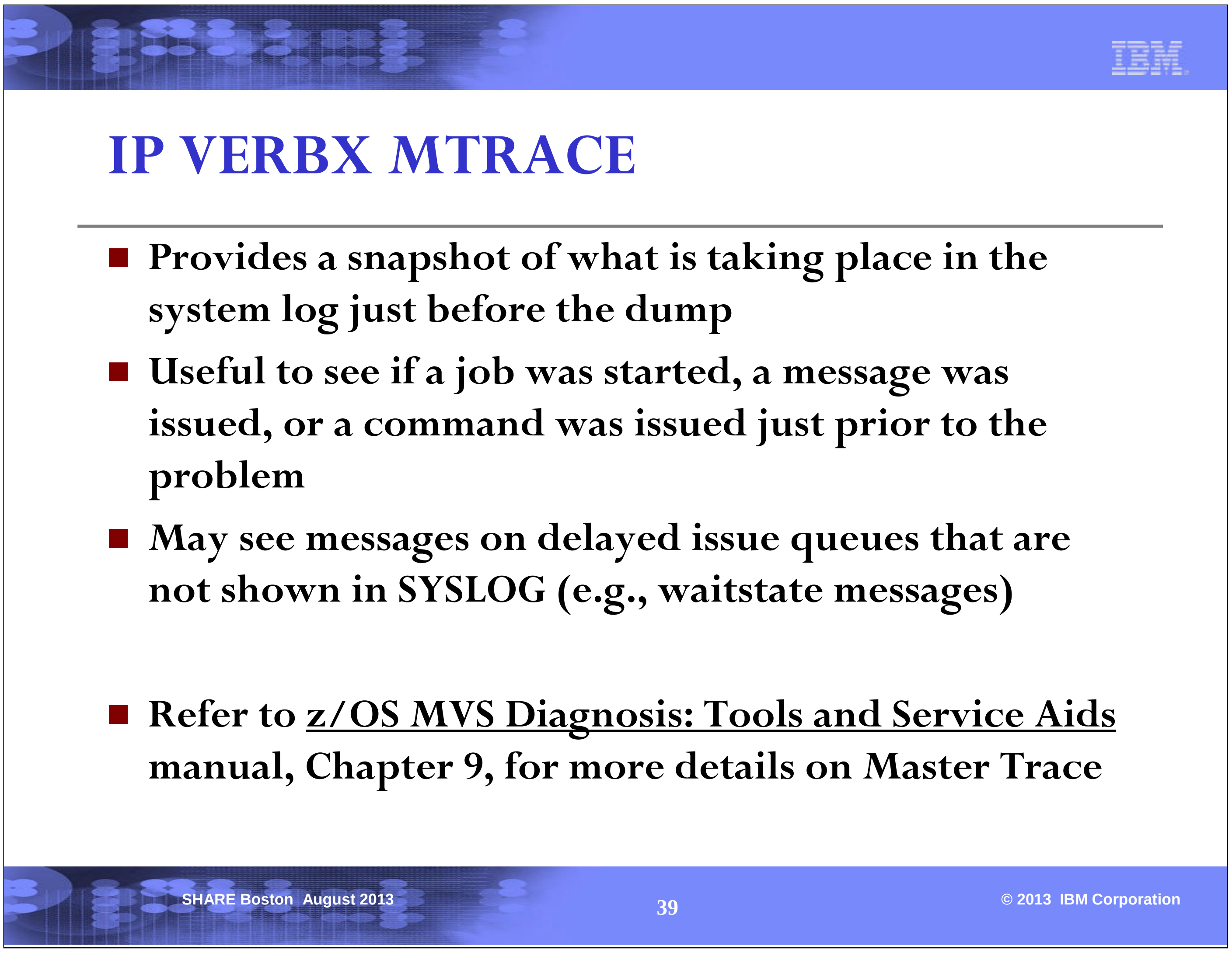
For an address that may point to a module or a common control block, the IPCS WHERE subcommand can be used. Note that the ASID parameter should be used if the address is in private storage.



How do I investigate recent errors or activity leading up to this dump?

- Review the recent system messages
- Review the recent LOGRECs
- Review the System Trace
- Analyze the TCB/RB structure
  - Applicable only if the problem is related to TCB(s) in a dumped address space

It is always a good practice to find out 'what led up to the problem'. There are a few areas in the dump that can help.

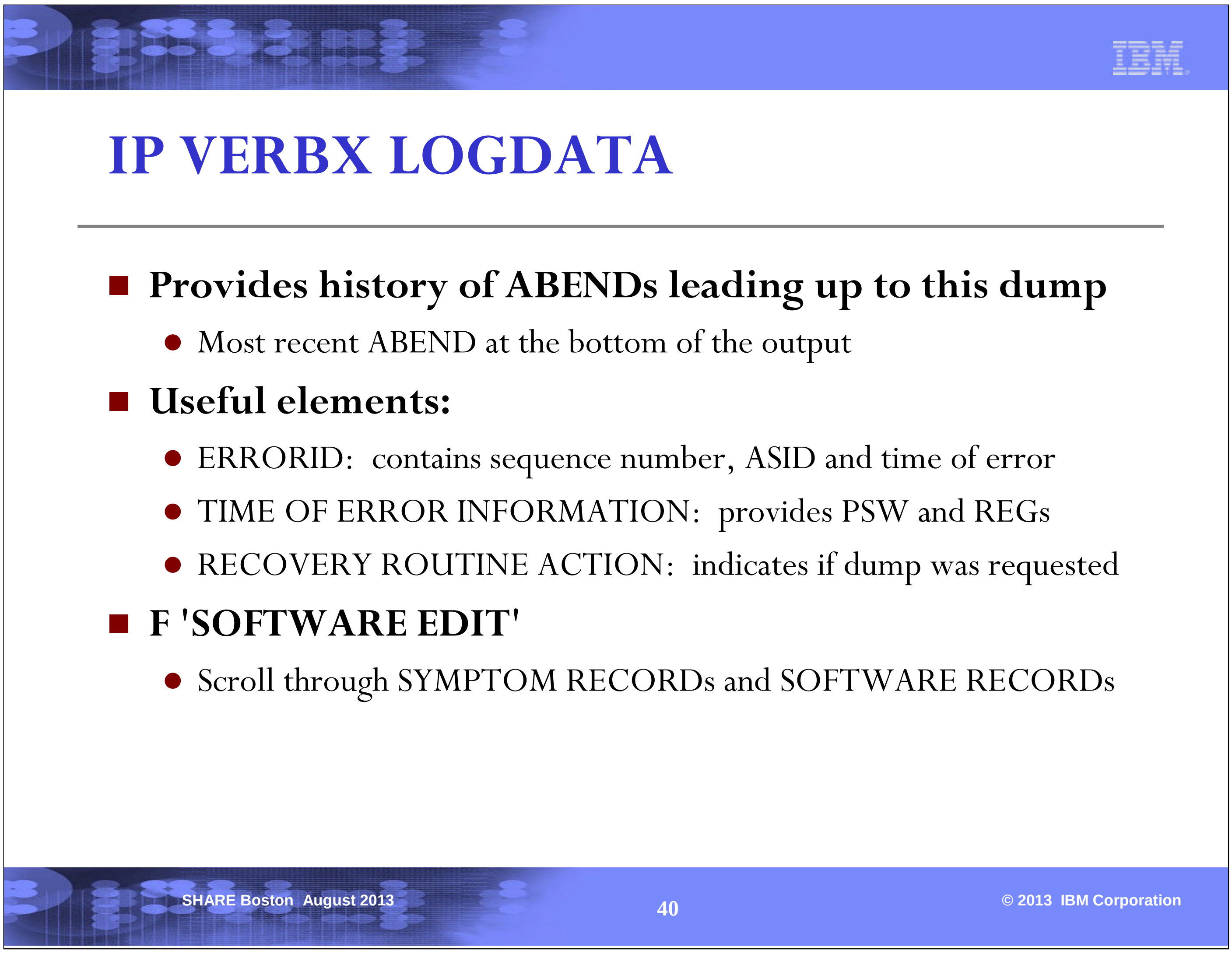


IP VERBX MTRACE

- Provides a snapshot of what is taking place in the system log just before the dump
- Useful to see if a job was started, a message was issued, or a command was issued just prior to the problem
- May see messages on delayed issue queues that are not shown in SYSLOG (e.g., waitstate messages)
- Refer to z/OS MVS Diagnosis: Tools and Service Aids manual, Chapter 9, for more details on Master Trace

SHARE Boston August 2013 39 © 2013 IBM Corporation

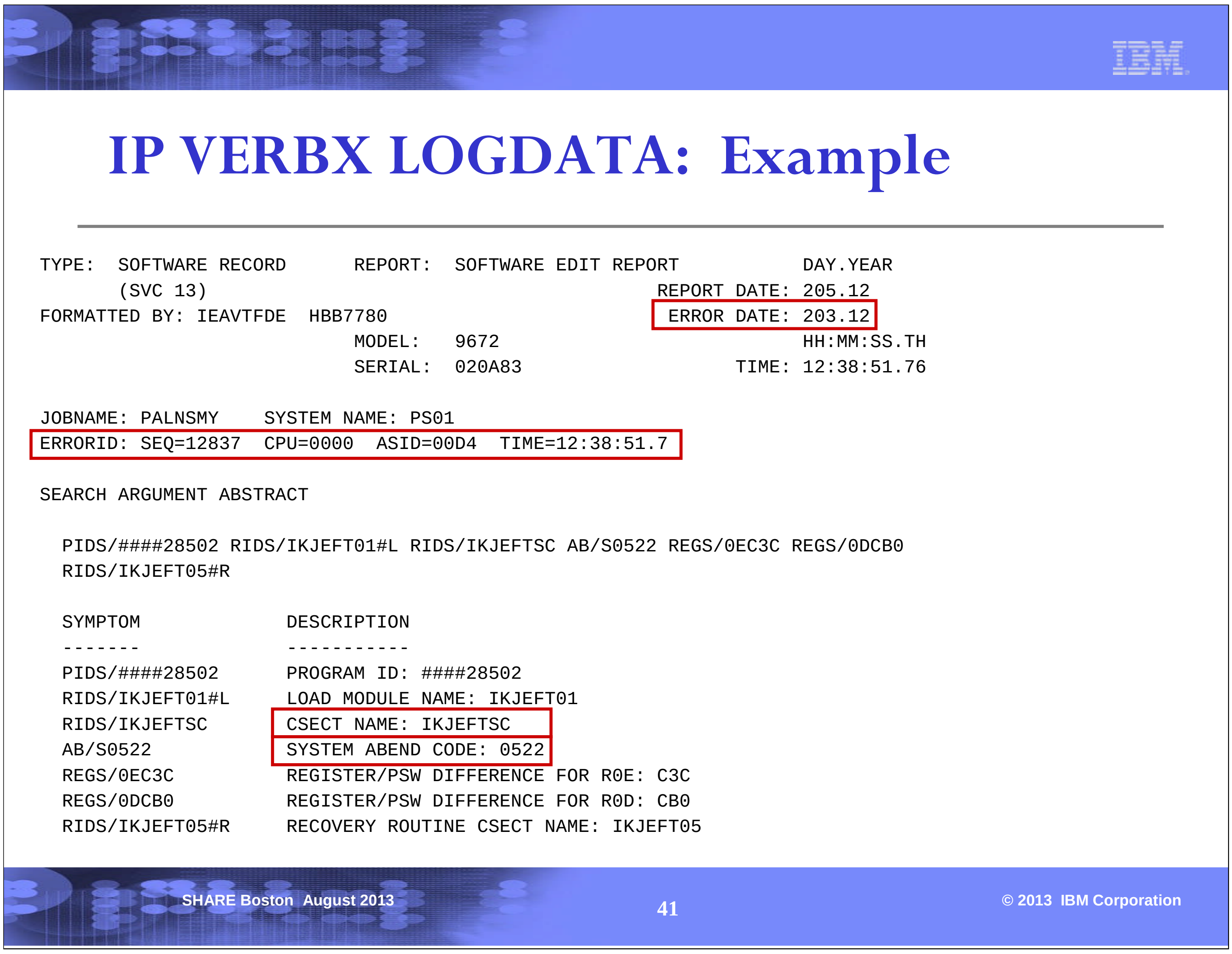
VERBX MTRACE displays the last messages that were issued to SYSLOG leading up to the dump, and may also give an indication of what jobs started just prior to the problem.



IP VERBX LOGDATA

- **Provides history of ABENDs leading up to this dump**
  - Most recent ABEND at the bottom of the output
- **Useful elements:**
  - ERRORID: contains sequence number, ASID and time of error
  - TIME OF ERROR INFORMATION: provides PSW and REGs
  - RECOVERY ROUTINE ACTION: indicates if dump was requested
- **F 'SOFTWARE EDIT'**
  - Scroll through SYMPTOM RECORDs and SOFTWARE RECORDs

VERBX LOGDATA is useful for reviewing the most recent ABENDs that occurred prior to the dump. The SOFTWARE RECORDs and SYMPTOM RECORDs are often of most value.



IP VERBX LOGDATA: Example

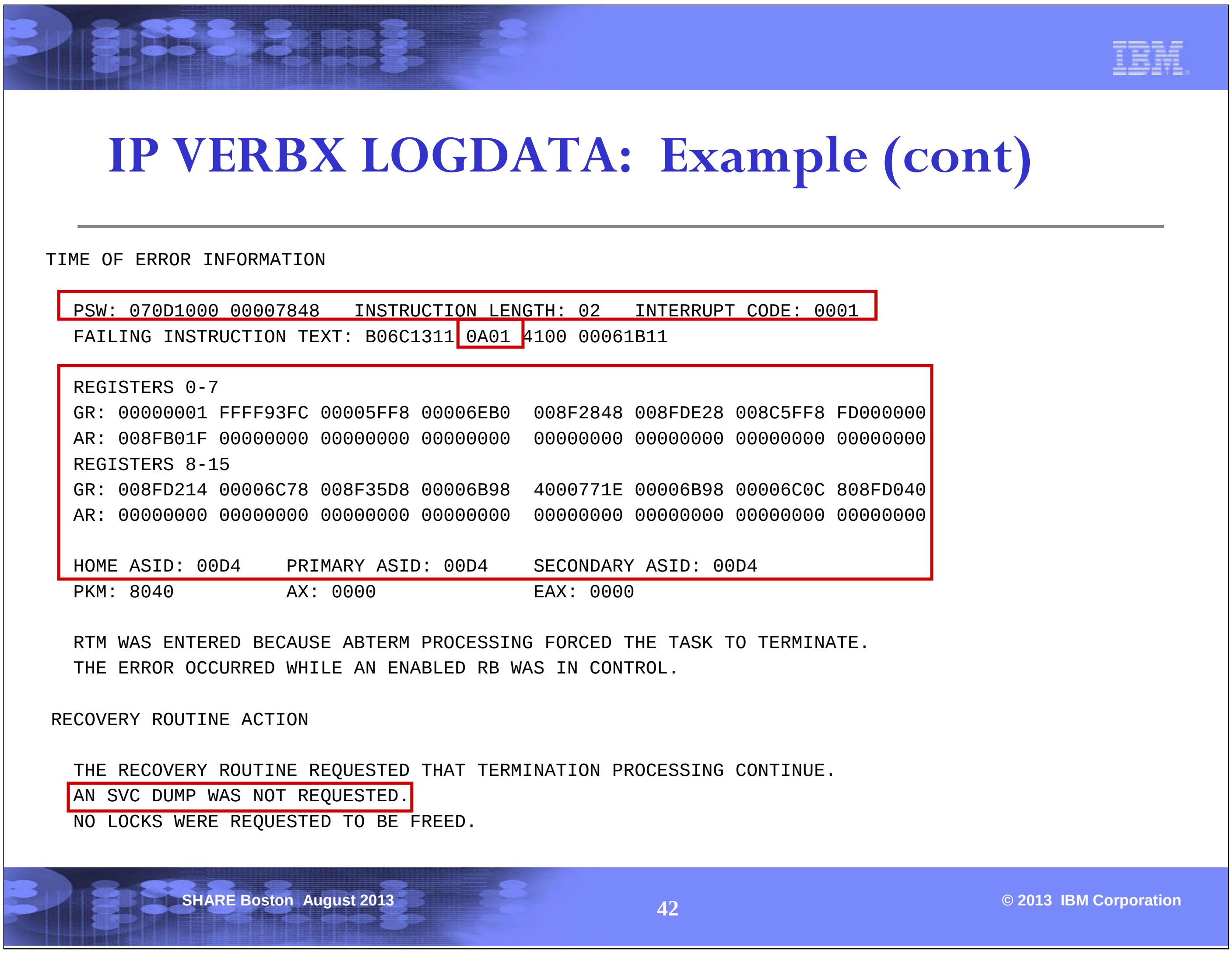
```
TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT           DAY.YEAR
       (SVC 13)                                         REPORT DATE: 205.12
FORMATTED BY: IEAVTFDE  HBB7780                          ERROR DATE: 203.12
                            MODEL:   9672                            HH:MM:SS.TH
                            SERIAL:  020A83                    TIME: 12:38:51.76

JOBNAME: PALNSMY    SYSTEM NAME: PS01
ERRORID: SEQ=12837  CPU=0000  ASID=00D4  TIME=12:38:51.7

SEARCH ARGUMENT ABSTRACT

  PIDS/####28502 RIDS/IKJEFT01#L RIDS/IKJEFTSC AB/S0522 REGS/0EC3C REGS/0DCB0
  RIDS/IKJEFT05#R

  SYMPTOM             DESCRIPTION
  -------             -----------
  PIDS/####28502      PROGRAM ID: ####28502
  RIDS/IKJEFT01#L     LOAD MODULE NAME: IKJEFT01
  RIDS/IKJEFTSC       CSECT NAME: IKJEFTSC
  AB/S0522            SYSTEM ABEND CODE: 0522
  REGS/0EC3C          REGISTER/PSW DIFFERENCE FOR R0E: C3C
  REGS/0DCB0          REGISTER/PSW DIFFERENCE FOR R0D: CB0
  RIDS/IKJEFT05#R     RECOVERY ROUTINE CSECT NAME: IKJEFT05
```

SEQ (sequence number): if sequence numbers are the same for multiple ABEND records, it indicates that this is the same ABEND being recorded by different recovery routines

ASID: ASID that encountered the error

TIME: Time the error occurred

This example is of an ABEND522 that occurred in TSO csect IKJEFTSC in asid(x'D4') jobname PALNSMY at 12:38:51 on Julian day 203 in the year 2012.



IP VERBX LOGDATA: Example (cont)

```
TIME OF ERROR INFORMATION

  PSW: 070D1000 00007848   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 0001
  FAILING INSTRUCTION TEXT: B06C1311 0A01 4100 00061B11

  REGISTERS 0-7
  GR: 00000001 FFFF93FC 00005FF8 00006EB0  008F2848 008FDE28 008C5FF8 FD000000
  AR: 008FB01F 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 008FD214 00006C78 008F35D8 00006B98  4000771E 00006B98 00006C0C 808FD040
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

  HOME ASID: 00D4     PRIMARY ASID: 00D4     SECONDARY ASID: 00D4
  PKM: 8040           AX: 0000               EAX: 0000

  RTM WAS ENTERED BECAUSE ABTERM PROCESSING FORCED THE TASK TO TERMINATE.
  THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL.

 RECOVERY ROUTINE ACTION

  THE RECOVERY ROUTINE REQUESTED THAT TERMINATION PROCESSING CONTINUE.
  AN SVC DUMP WAS NOT REQUESTED.
  NO LOCKS WERE REQUESTED TO BE FREED.
```

The failing instruction was an SVC 1 (x’0A01’) Wait that was issued by PSW address x’7846’. Note that the PSW address x’7848’ points after the failing instruction, so we had to back up by the INSTRUCTION LENGTH (02) to find the failing PSW address x’7846’ and failing instruction (x’0A01) in the FAILING INSTRUCTION TEXT.

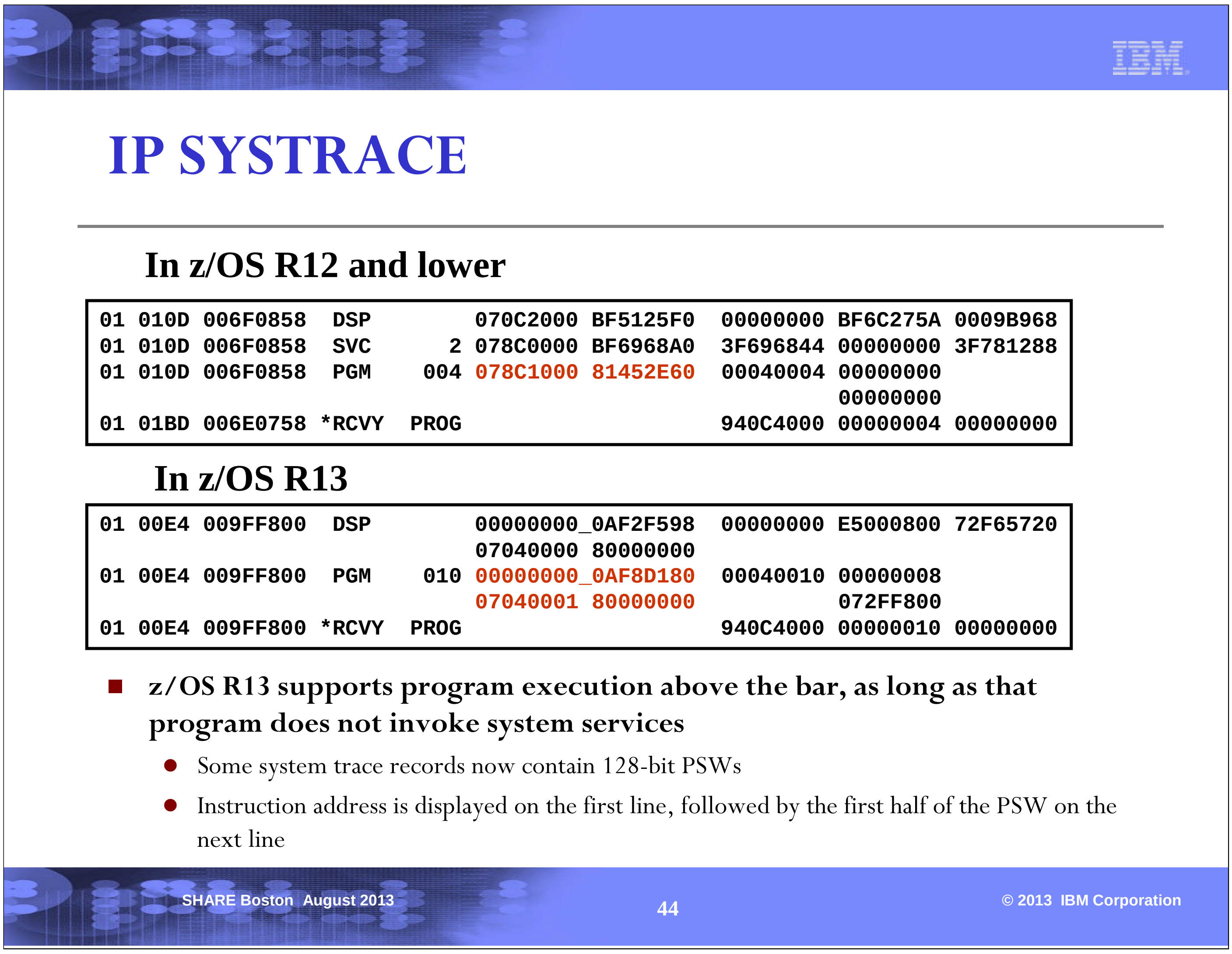
The registers displayed are the registers at the time of the failure (ABEND522). An SVC dump was not requested by recovery for this ABEND522.

IP SYSTRACE

- **System Trace activity at time of dump**
- **Examples:**
  - IP SYSTRACE ASID(x'nn') TIME(LOCAL)
    - Formats only trace records associated with the requested ASID
  - IP SYSTRACE ALL TIME(LOCAL)
    - Formats trace records for all active address spaces in the system

- **Refer to <u>z/OS MVS Diagnosis: Tools and Service Aids</u> manual, Chapter 8, for more details on System Trace**

The TIME(LOCAL) parameter converts the time in SYSTRACE to local time. The default is raw hex timestamps.



IP SYSTRACE

In z/OS R12 and lower

```
01 010D 006F0858   DSP        070C2000 BF5125F0  00000000 BF6C275A 0009B968
01 010D 006F0858   SVC      2 078C0000 BF6968A0  3F696844 00000000 3F781288
01 010D 006F0858   PGM    004 078C1000 81452E60  00040004 00000000
                                                          00000000
01 01BD 006E0758 *RCVY   PROG                    940C4000 00000004 00000000
```

In z/OS R13

```
01 00E4 009FF800   DSP        00000000_0AF2F598  00000000 E5000800 72F65720
                              07040000 80000000
01 00E4 009FF800   PGM    010 00000000_0AF8D180  00040010 00000008
                              07040001 80000000           072FF800
01 00E4 009FF800 *RCVY   PROG                    940C4000 00000010 00000000
```

- **z/OS R13 supports program execution above the bar, as long as that program does not invoke system services**
  - Some system trace records now contain 128-bit PSWs
  - Instruction address is displayed on the first line, followed by the first half of the PSW on the next line

The system trace table consists of many different kinds of entries. Prior to z/OS V1 R13, all PSWs in the system trace entries are 64-bit (scrunched). z/OS V1 R13 is the first release to support program execution above the 2G bar, as long as the program does not invoke system services. Since the instruction address can now be greater than 31 bits, some of the system trace entries have been changed to contain 128-bit PSWs. Note that the 128-bit PSW is displayed in 2 lines, but not in the order of bit 0 to bit 127. The instruction address is displayed first, then followed by the first half of the PSW in the next line.

IP SUMM FORMAT ASID(x'nn')

- Formats the key control blocks of an address space
  - First scroll down to bottom to see the TCB summary

```
 * * * *   T C B   S U M M A R Y   * * * *

JOB CONSOLE  ASID 000B ASCB 00FA8080 FWDP 00000000 BWDP 00000000
00000008
    TCB AT    CMP      NTC      OTC      LTC      TCB      BACK
   008FE050 00000000 00000000 00000000 008FF890 008FD0D0 00000000
   008FD0D0 00000000 00000000 008FE050 00000000 008FF890 008FE050
   008FF890 00000000 008FD0D0 008FE050 008F0938 008FF2A0 008FD0D0
   008FF2A0 00000000 00000000 008FF890 008F92F0 008F9E88 008FF890
   008F9E88 00000000 008FF2A0 008FF890 00000000 008F9CF0 008FF2A0
   .....
   ...
   008FABD0 940C4000 00000000 008F5170 00000000 00000000 00647E88
```

Lines omitted here

non-zero completion code (ABEND0C4)

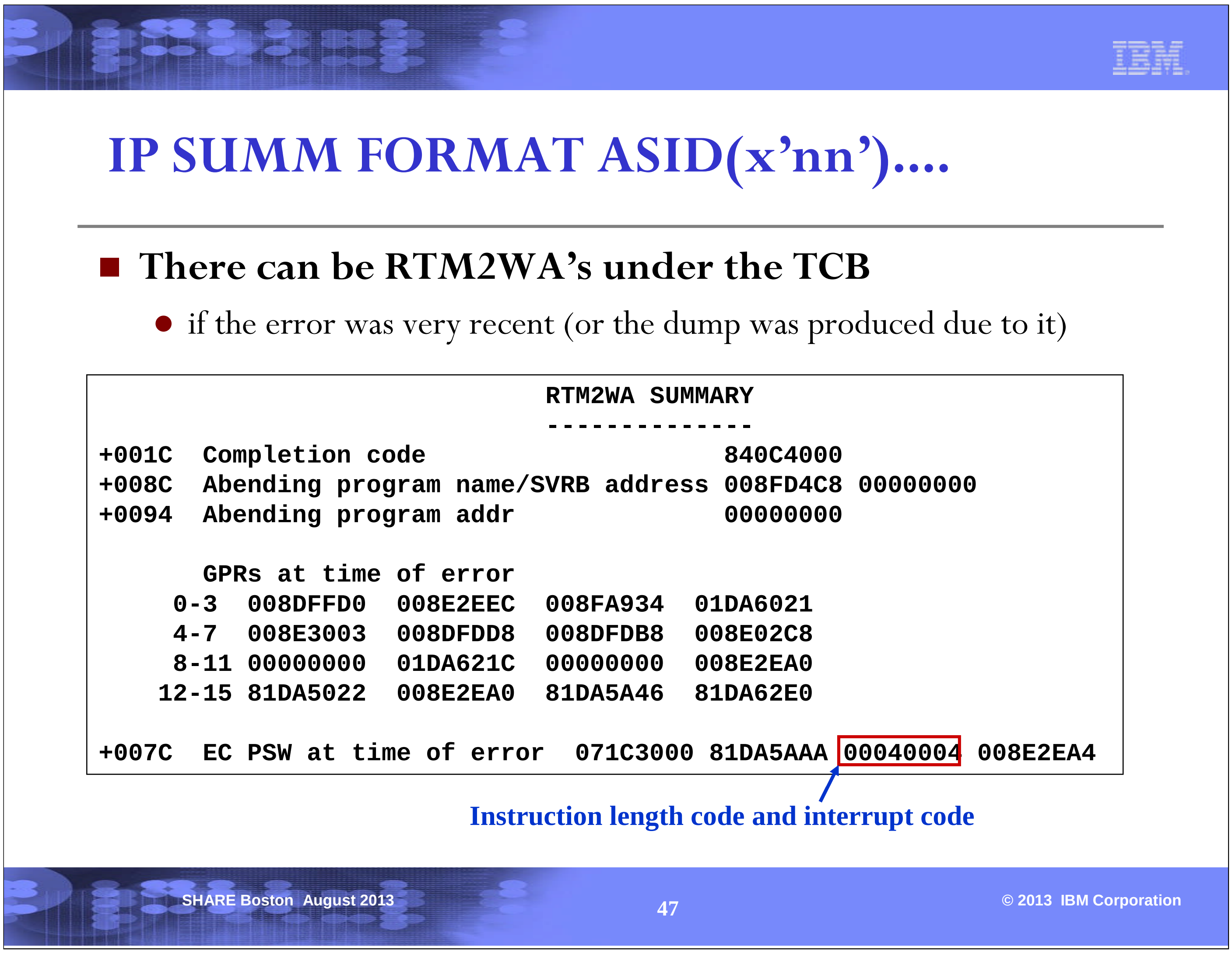
The SUMMARY subcommand can be used to format the control blocks of an address space. The TCBs in the address space should be investigated for any recent errors. At the bottom of the SUMM FORMAT output is the TCB summary. Note any TCBs with non-zero completion code under the CMP field.

IP SUMM FORMAT ASID(x'nn')....

- Find the TCB(s) with non-zero completion code
  - note: completion code can be residual

```
TCB: 008FABD0
    +0000  RBP...... 008FD7A8  PIE...... 00000000  DEB...... 00000000
    +000C  TIO...... 008E7000  CMP...... 940C4000  TRN...... 40000000
    +0018  MSS...... 7FF156F0  PKF...... 10        FLGS..... 01000000
    +0022  LMP...... FF        DSP...... FF        LLS...... 00000000
    +0028  JLB...... 00000000  JPQ...... 00000000
  GENERAL PURPOSE REGISTER VALUES
      0-3  00F8538D  00C7BEB0  00C7BEB0  01B63688
      4-7  02FE9CE8  00000018  01E44085  01B63568
     8-11  00C7BCE0  81E43086  00000000  00000000
    12-15  00C7BE88  01B63B38  81E431B6  00000000
```

Once you found a TCB with non-zero completion code, you can issue a FIND of 'TCB: 00xxxxxx' from the top of the output to find the TCB. The error under this TCB may be the one causing the dump to be taken. Or it can be a residual completion code.



IP SUMM FORMAT ASID(x'nn')....

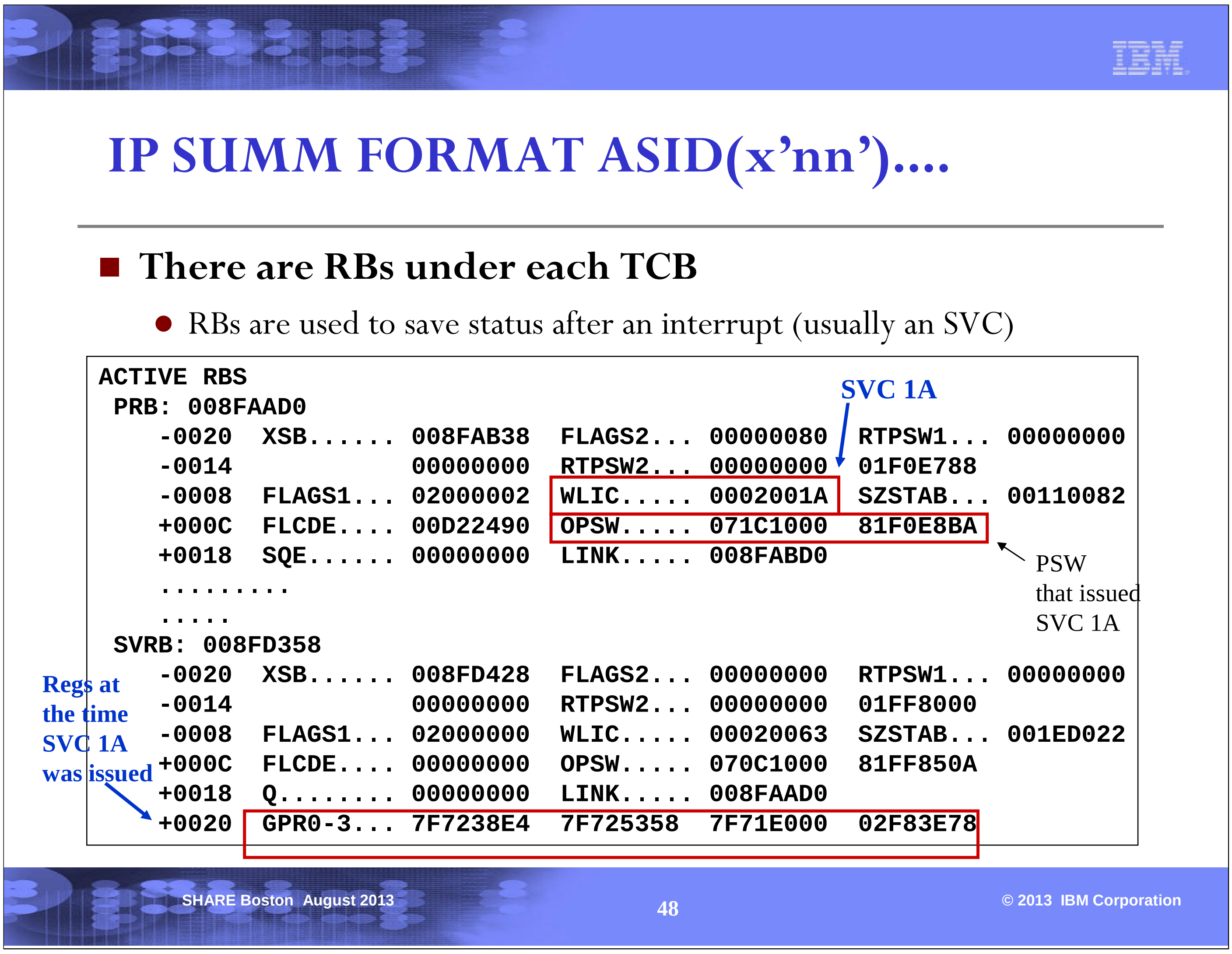
- There can be RTM2WA's under the TCB
  - if the error was very recent (or the dump was produced due to it)

```
                                 RTM2WA SUMMARY
                                 --------------
+001C  Completion code                      840C4000
+008C  Abending program name/SVRB address 008FD4C8 00000000
+0094  Abending program addr               00000000

       GPRs at time of error
     0-3  008DFFD0  008E2EEC  008FA934  01DA6021
     4-7  008E3003  008DFDD8  008DFDB8  008E02C8
     8-11 00000000  01DA621C  00000000  008E2EA0
    12-15 81DA5022  008E2EA0  81DA5A46  81DA62E0

+007C  EC PSW at time of error  071C3000 81DA5AAA 00040004 008E2EA4
```

Instruction length code and interrupt code

If the TCB is going through recovery processing for the error in the completion code, you will find a RTM2WA (RTM2 Work Area) after the TCB. The above shows the RTM2WA Summary which contains the PSW at time of error and the registers, as well as the instruction length code, interrupt code, and the translation exception address (not applicable in this case since the interrupt code represents a protection exception).

The Translation Exception Address is the address that caused a PIC 10,11,38,39,3A or 3B.



IP SUMM FORMAT ASID(x’nn’)....

- **There are RBs under each TCB**
  - RBs are used to save status after an interrupt (usually an SVC)

```
ACTIVE RBS
 PRB: 008FAAD0
    -0020  XSB...... 008FAB38  FLAGS2... 00000080  RTPSW1... 00000000
    -0014            00000000  RTPSW2... 00000000  01F0E788
    -0008  FLAGS1... 02000002  WLIC..... 0002001A  SZSTAB... 00110082
    +000C  FLCDE.... 00D22490  OPSW..... 071C1000  81F0E8BA
    +0018  SQE...... 00000000  LINK..... 008FABD0
    .........
    .....
 SVRB: 008FD358
    -0020  XSB...... 008FD428  FLAGS2... 00000000  RTPSW1... 00000000
    -0014            00000000  RTPSW2... 00000000  01FF8000
    -0008  FLAGS1... 02000000  WLIC..... 00020063  SZSTAB... 001ED022
    +000C  FLCDE.... 00000000  OPSW..... 070C1000  81FF850A
    +0018  Q........ 00000000  LINK..... 008FAAD0
    +0020  GPR0-3... 7F7238E4  7F725358  7F71E000  02F83E78
```

SVC 1A

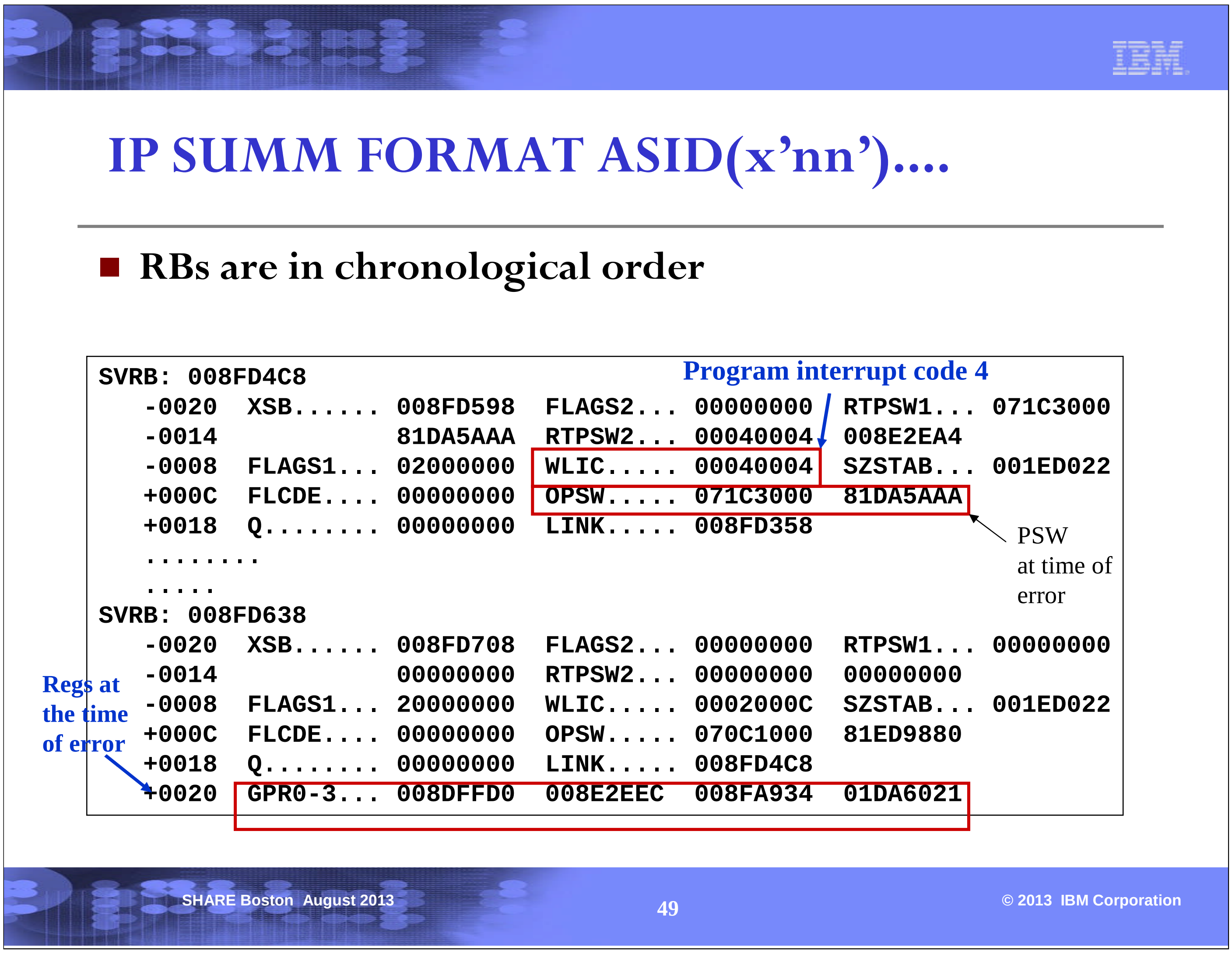
PSW that issued SVC 1A

Regs at the time SVC 1A was issued

The RBs are used to save status (PSW and Registers) after an interrupt, and can be used to show the recent activity of the TCB.

In the SUMM FORMAT output, RBs are listed in chronological order: oldest RB at the top, most recent RB at the bottom. Note that each TCB has RBs under it, so make sure that you are looking at the right ones. Check RBLINK field (offset x’1C’) of the first RB, it contains the TCB address. RBLINK in subsequent RBs points backwards to the previous RB.

In this example, the first RB indicates that an SVC x’1A’ (LOCATE SVC) was issued at 1F0E8BA. The WLIC field contains the instruction length code and the interrupt code, and an SVC instruction has a length of 2 bytes. The registers at the time of the SVC x’1A’ were saved in the next RB (an SVRB). Then an SVC x’63’ (DYNAMIC ALLOCATION SVC) was issued from 1FF850A. The registers at the time of the SVC x’63’ were saved in the next SVRB (see next page).



IP SUMM FORMAT ASID(x'nn')....

- RBs are in chronological order

```
SVRB: 008FD4C8
    -0020  XSB...... 008FD598  FLAGS2... 00000000  RTPSW1... 071C3000
    -0014            81DA5AAA  RTPSW2... 00040004  008E2EA4
    -0008  FLAGS1... 02000000  WLIC..... 00040004  SZSTAB... 001ED022
    +000C  FLCDE.... 00000000  OPSW..... 071C3000  81DA5AAA
    +0018  Q........ 00000000  LINK..... 008FD358
    ........
    .....
SVRB: 008FD638
    -0020  XSB...... 008FD708  FLAGS2... 00000000  RTPSW1... 00000000
    -0014            00000000  RTPSW2... 00000000  00000000
    -0008  FLAGS1... 20000000  WLIC..... 0002000C  SZSTAB... 001ED022
    +000C  FLCDE.... 00000000  OPSW..... 070C1000  81ED9880
    +0018  Q........ 00000000  LINK..... 008FD4C8
    +0020  GPR0-3... 008DFFD0  008E2EEC  008FA934  01DA6021
```

Then under the processing of the SVC x'63', a program check occurred (protection exception) at 1DA5AAA. The WLIC field has 0004004. This does not represent an SVC interrupt because the instruction length is 4. It was a program interrupt. The registers at the time of error were saved in the next SVRB. The second SVRB in the picture represents RTM processing after the PIC 4.

Note that eventually you will reach the bottom RB (the most recent RB). This most recent RB is called the Top RB and is pointed to by TCBRBP (sorry, it is confusing, the Top RB is at the bottom). The registers of the Top RB are saved in the TCB.

RB in IPCS SUMM FORMAT ASID(x'nn')

In z/OS R13

```
SVRB: 009FD548
   -0020  XSB...... 7FFF9338  FLAGS2... 00        RTPSW1... 070C0001
   -0014            89F8D186  RTPSW2... 0006003B  7FFFFBD0
   -0008  FLAGS1... 02000000  WLIC..... 0006003B
   +0000  RSV...... 00000000  00000000            SZSTAB... 001ED022
   +000C  CDE...... 00000000  OPSW..... 070C0001  89F8D186
```

```
XSB: 7FFF9338
   +0000  XSB...... XSB       LINK..... 00000000  XLIDR.... 00000000
   +0014  XLAS..... 00000000  TKN...... 0000      ASD...... 0000
......
....   Lines omitted here
...
+00D4  AX....... 0000      PASID.... 00D4
+00D8  BEA...... 00000000  09F8BF52
+00E0  PSW16.... 07040001  80000000  00000000  09F8D186
```

- RBOPSW still exists but it is not relied upon
- XSBOPSW16 contains the official 128-bit PSW

The RB contains a field called RBOPSW, which is used by z/OS to save the interrupted PSW of a TCB. This PSW is 64-bit and will still exist in z/OS V1 R13. It will be maintained by z/OS, but it will not be not used by z/OS to re-dispatch the TCB. The official 128-bit PSW is now in the XSB. In a dump, you should usually find that the RBOPSW and XSBOPSW16 are similar.

Session 14181
z/OS Debugging: *Old Tricks* *for* *New Dogs*

Thank You

Your comments will be greatly appreciated

SHARE Boston August 2013