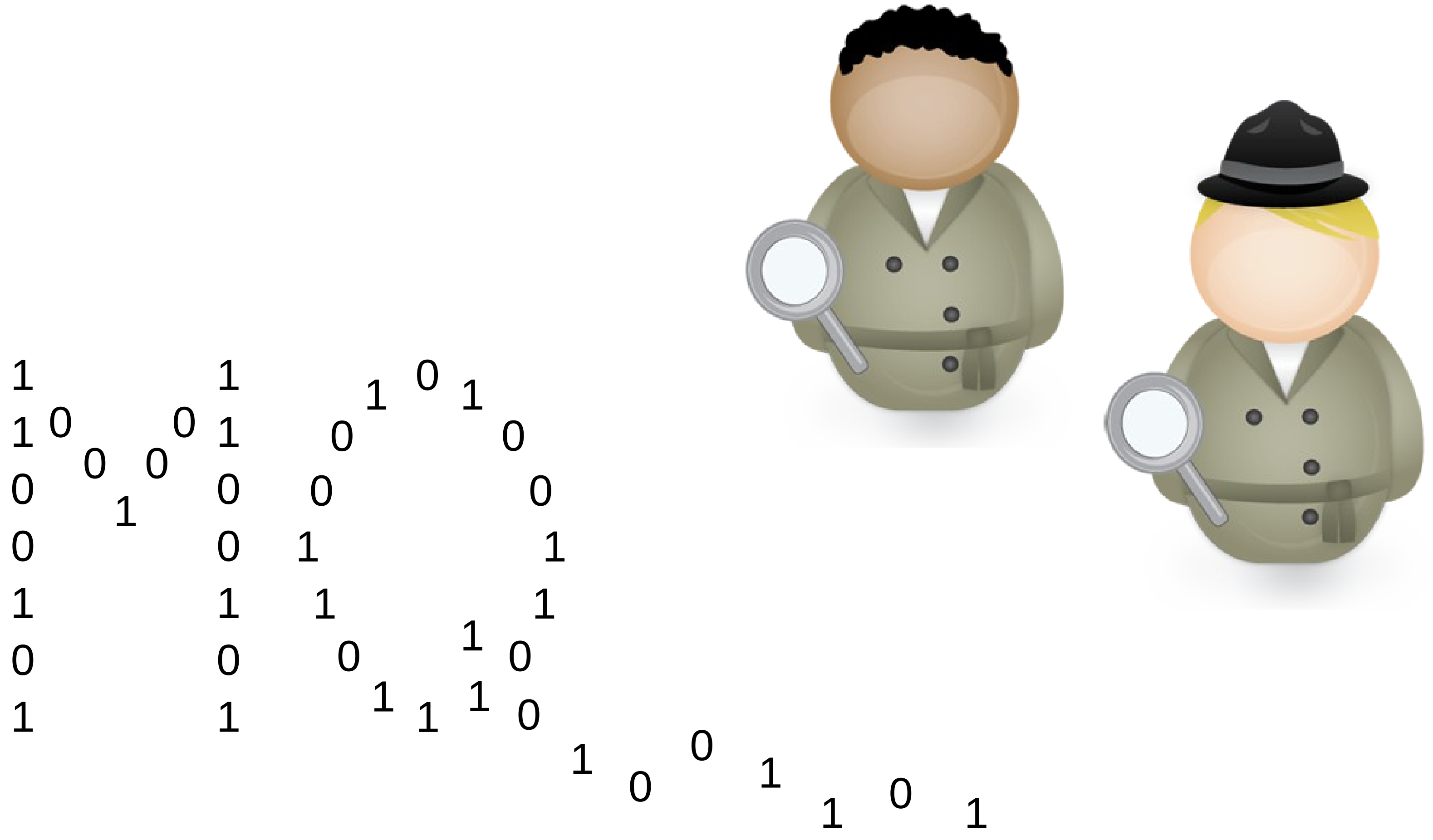
Common problems and problem determination for MQ z/OS

Dirk Marski, IBM
MQ for z/OS Support

Agenda

- MQ Detectives – Problem Determination
- "My application failed".
  - Gathering available information.
  - Creating additional diagnostic data.
- "My message is missing".
  - Message tracking techniques.
    - Locating a message in a simple system.
  - Advanced message tracking.
    - Identifying message delivery routes.
    - Delayed messages.
- How to avoid problems.

MQ Detectives – Problem Determination

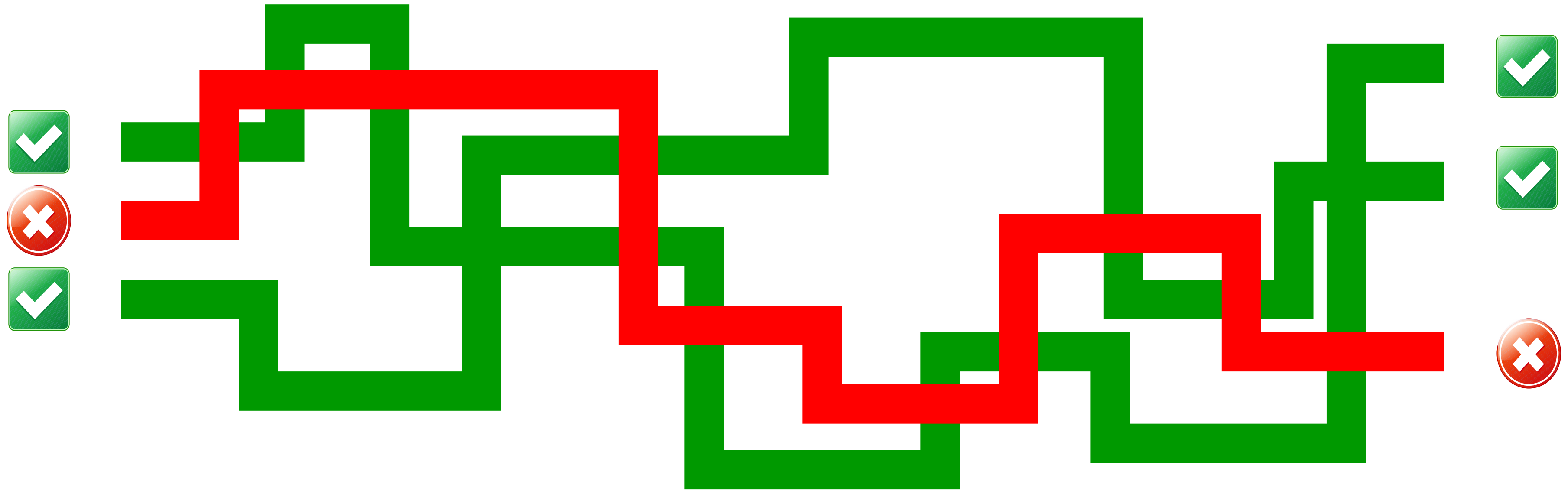


Problem Determination Methodology

- Problems are different on many levels:
  - How they manifest
  - The circumstances under which they occur
  - The ways in which they can be addressed
- The way of determining root cause is fairly common:
  - The problem occurred
    - Don't disturb the crime scene
    - Bag and tag the evidence
  - Ask questions
  - Follow the evidence
  - Build a hypothesis that is supported by the facts

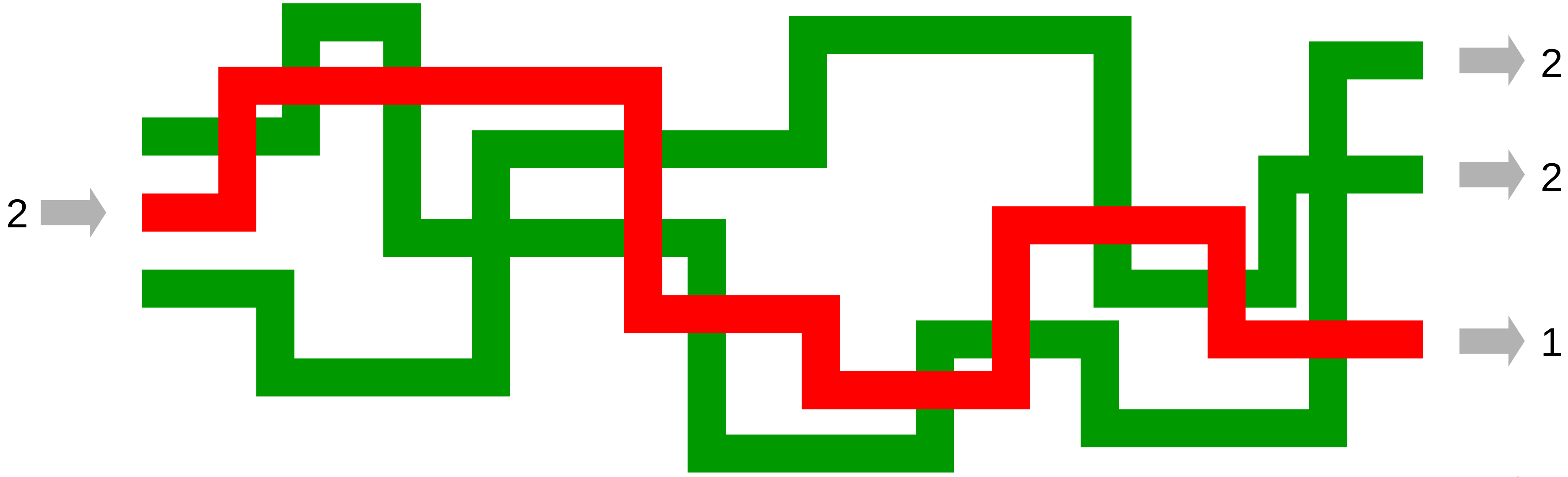
Problem Determination Methodology – cont'd

- Problem path or sequence of events → “The time line”
  - Many options, some are normal, one is the error path



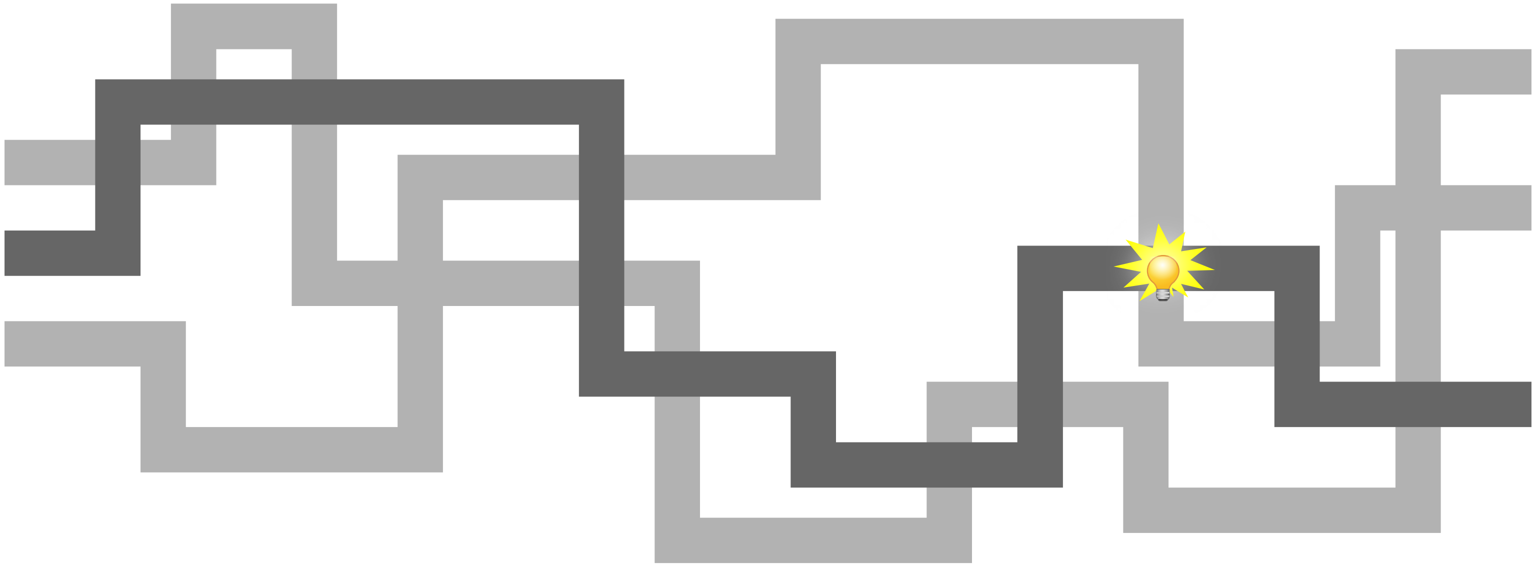
Problem Determination Methodology – cont'd

- 2 types of scenarios:
  1) The outcome is known (for instance an abend)
  2) The trigger is known (for instance putting a given message to a particular queue)



Problem Determination Methodology – cont'd

- Starting point → symptoms!

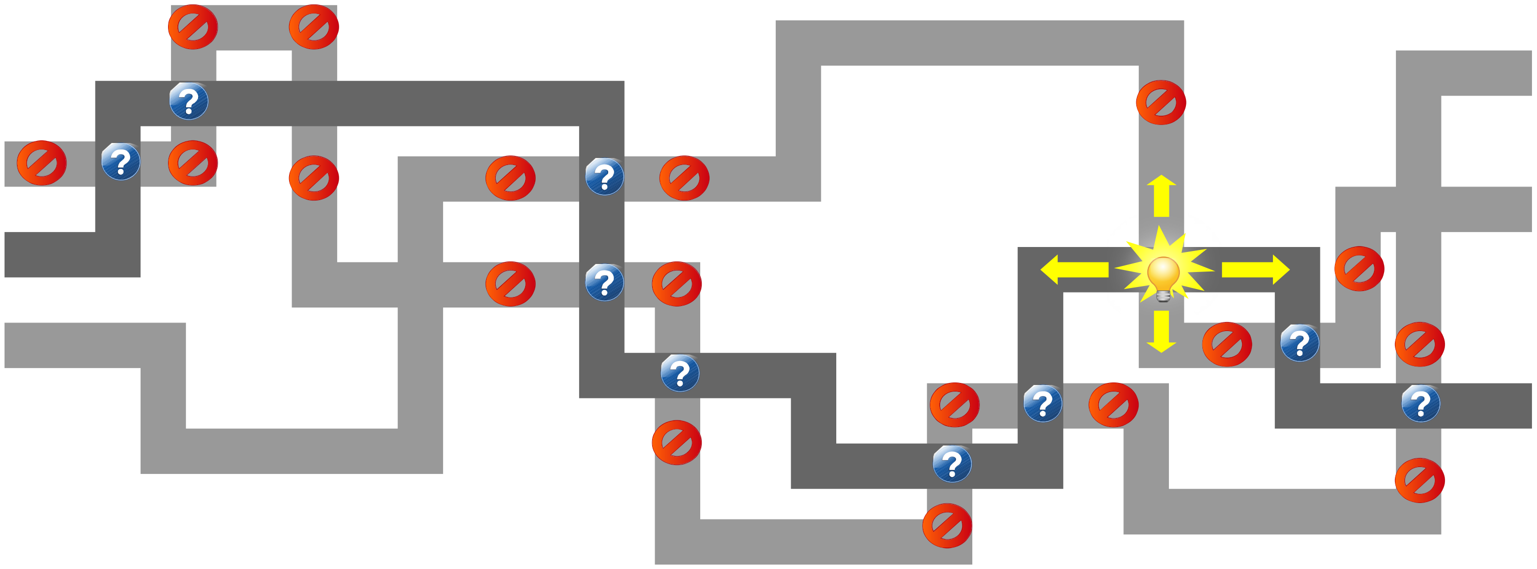


Problem Determination Methodology – cont'd

- Symptoms – what can you see?
  - “My message is missing” ⚠
  - “My application did not receive a message on the reply queue within 10 minutes” ✓
  - “The queue manager hangs” ⚠
  - “The queue manager is not responding to console commands” ✓

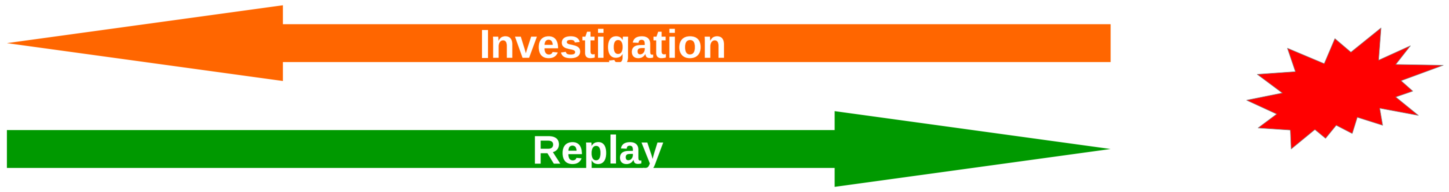
Problem Determination Methodology – cont'd

- Experience:
  - Identify blind alleys early on
  - Reveal new paths you would not have considered
  - Prioritise what to spend time on



Problem Determination Methodology – cont'd

- Story
  - Following the trail backwards is easy, but difficult to communicate



  - It is all about telling the story forwards
  - Telling it both ways is a good way to validate every aspect has been understood

Problem Determination Methodology – cont'd

- Questions:
  - Has something changed in your system?
  - Look at the wider environment
  - Has this worked in the past?
  - Was there anything unusual at the time of the problem (high workload, network blip, system outage ...)
- Spend time looking at the possibilities before doing any deep digging down a given path
- Look up every now and then to see if this is the right path to go down

Problem Determination Methodology – cont'd

- Insufficient documentation?
  - Think about what the information would provide before requesting it
- Be prepared!
  - Install CCTV and alarm
    - set trace, monitoring, dump capture and suppression
  - Know what your system looks like normally
    - Spot the difference when something has gone wrong

“My application failed”

Ask the user and application owner

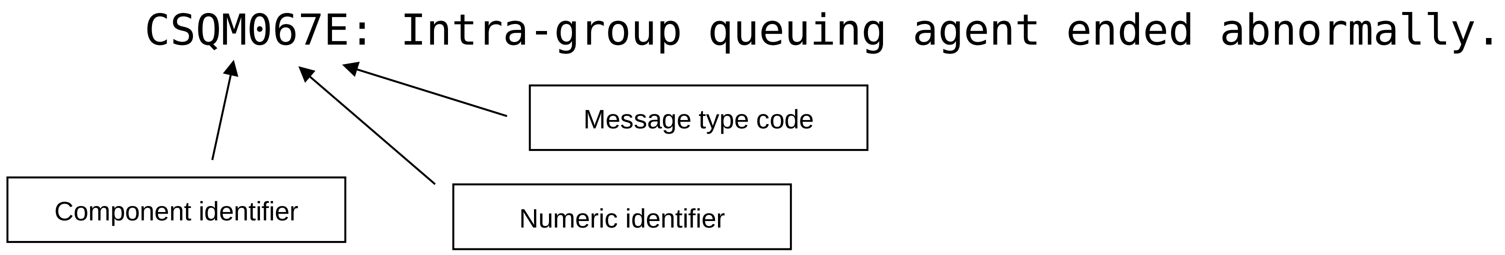
- What were they doing?
  - Which application, queue manager and queue?
  - Was this normal processing, or something unusual?
- What went wrong?
  - Get specific details.
  - Any error messages?
- What was the expected result?
- When did it happen?
  - Only once
  - Repeatedly over a period
  - Still occurring

Application symptoms – bag it and tag it!

- MQ provides details about failures to the application
  - Specific reason codes
- Check application error logs
  - Detailed error reports are a big help
    - “Application failed” - Unhelpful
    - “Error opening queue with completion code 2” – Slightly better
    - “MQOPEN failed with reason 2059 for APP1.REPLY” – Good
- Applications can have multiple components
  - Web page – servlet – EJB – JMS – MQ API
  - Errors may be reported in several places

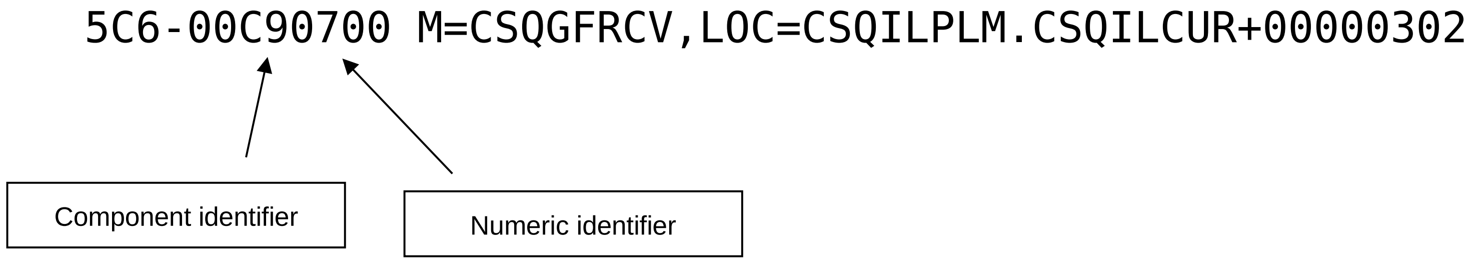
MQ error reporting

- MQ MSTR and CHIN tasks provide diagnostics for errors
  - Messages in joblog



```
CSQM067E: Intra-group queuing agent ended abnormally.
```

- Task abends
  - Abend code x’5C6’ or x’6C6’
  - Reason code identifies cause



```
5C6-00C90700 M=CSQGFRCV,LOC=CSQILPLM.CSQILCUR+00000302
```

GTF trace

- MQ uses z/OS GTF trace facility for diagnostic trace.
- API trace and internal trace
  - 5E9 – API entry
  - 5EA – API exit
  - 5EE – Internal trace
- Trace data written to wrapping dataset
- IPCS formatting required to produce readable output.

GTF trace cont’d

- Start GTF

```
START GTF.DB
   £HASP100 GTF.DB ON STCINRDR
   £HASP373 GTF.DB STARTED
  *01 AHL100A SPECIFY TRACE OPTIONS
R 01,TRACE=JOBNAMEP,USRP
   TRACE=JOBNAMEP,USRP
   IEE600I REPLY TO 01 IS;TRACE=JOBNAMEP,USRP
  *02 ALH101A SPECIFY TRACE EVENT KEYWORDS - JOBNAME=,USR=
R 02,JOBNAME=(MQ11MSTR,MQAPP1),USR=(5E9,5EA)
   JOBNAME=(MQ11MSTR,MQAPP1),USR=(5E9,5EA)
   IEE600I REPLY TO 02 IS;JOBNAME=(MQ11MSTR,MQAPP1),USR=(5E9,5EA)
  *03 ALH102A CONTINUE TRACE DEFINITION OR REPLY END
R 03,END
   END
   IEE600I REPLY TO 03 IS;END
   AHL103I TRACE OPTIONS SELECTED-USR=(5E9,5EA)
   AHL103I JOBNAME=(MQ11MSTR,MQAPP1)
  *04 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 04,U
   U
   IEE600I REPLY TO 04 IS;U
   AHL031I GTF INITIALIZATION COMPLETE
```

GTF trace cont'd

- Start MQ Trace

```
+MQ11 START TRACE(G)CLASS(3) DEST(GTF)
```

  – All Entry and Exit

```
+MQ11 START TRACE(G)CLASS(2) DEST(GTF)
```

  – Only when exit reason is not MQRC_NONE

- Other MQ trace control

```
+MQ11 DISPLAY TRACE ...

+MQ11 ALTER TRACE ...

+MQ11 STOP TRACE ...
```

GTF trace cont'd

- Example output

```
USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
PUTONE
    Thread... 004C2B10  Userid... CICSUSER  pObjDesc. 106B2010
    pMsgDesc. 106B20B8  pPMO..... 106B2200  BufferL.. 00000064
    pBuffer.. 106A0578  RSV1..... 00000000  RSV2..... 00000000
    RSV3..... 116BC830  CompCode. 00000002  Reason... 000007FB
    C9E8C1E8  C5C3C9C3  AA8E8583  76270484  | IYAYECIC..ec...d |
    D4D8E3E3  0000048C  00000000  00000000  | MQTT............ |
    00000000  1910C7C2  C9C2D4C9  E8C14BC9  | ......GBIBMIYA.I |
    C7C3E2F2  F0F48E85  83762979  00010000  | GCS204.ec..`.... |
MQRC_OBJECT_TYPE_ERROR

               GMT-01/30/05 14:42:08.412678    LOC-01/30/05 14:42:08.412678

USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
 +0000  D6C44040  00000001  00000000  C2404040  | OD  ........B    |
 ...
 +00A0  00000000  00000000                      | ........         |
```

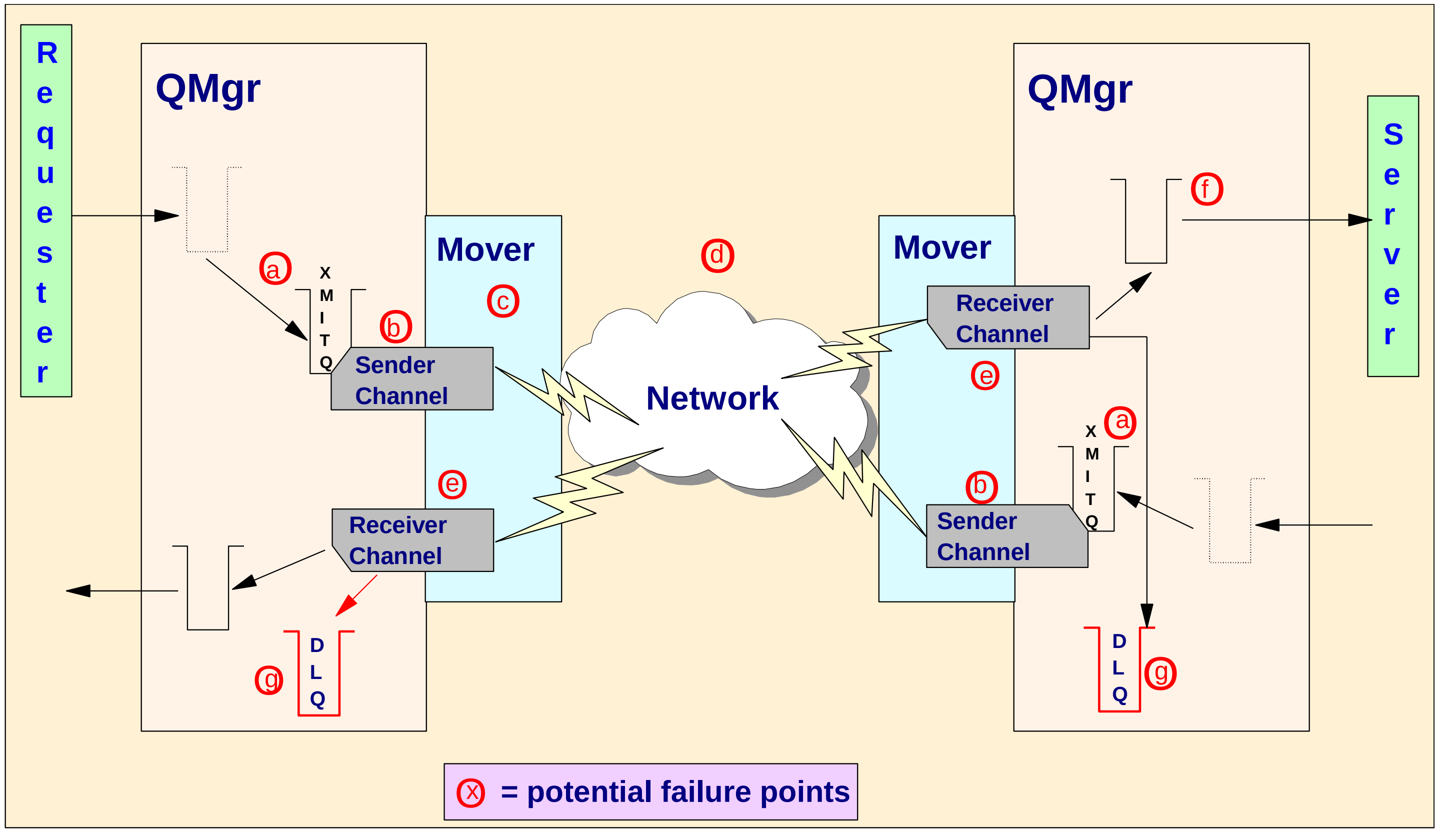
Capturing a dump

- z/OS system dumps are an important tool for capturing system state at the time of an error.
- Dump may have already been captured.
  - MQ 5C6 abends
  - Application requested dump
  - Other z/OS components
- Several methods to generate a dump for a failure
  - Console DUMP command
  - SLIP trap
  - RECOVER QMGR(MQRD,2051,1)
- MQ Dump formatters CSQWDPRD and CSQXDPRD

“My message is missing”
Message tracking techniques

Where might it have gone wrong?

- A simple request/reply application



Should the message still be in MQ?

- There are valid reasons why a message could be removed from MQ.
  - Was the MQPUT successful?
  - Did the application commit?
  - Is the message non-persistent?
    - Queue manager restart
    - Channel failure
    - Read ahead
  - Did the message have expiry set?
  - Clear queue

MQ Commands

- Command interfaces to inquire on MQ object status
  - MQSC – Text format commands
  - PCF – Programmable format, useful for monitoring applications
  - Information also obtainable via tools
    - MQExplorer
    - MQ Operations and Control ISPF panels
- Display object commands show object attributes
    - E.g. DISPLAY QUEUE(APP1.INPUT) MAXDEPTH
- Display status commands show current state information
    - E.g. DISPLAY QSTATUS(APP1.INPUT) CURDEPTH

MQ Commands cont'd

- DISPLAY QSTATUS

```
+MQ11 DISPLAY QSTATUS(APP1.INPUT) ALL
CSQM293I +MQ11 CSQMDRTC 1 QSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I +MQ11 CSQMDRTC  DISPLAY QSTATUS DETAILS
QSTATUS(APP1.INPUT)
TYPE(QUEUE)
OPPROCS(1)
IPPROCS(0)
CURDEPTH(4)
UNCOM(NO)
MONQ(HIGH)
QTIME(6639576,9403795)
MSGAGE(7)
LPUTDATE(2011-07-30)
LPUTTIME(21.15.57)
LGETDATE(2011-07-30)
LGETTIME(21.16.00)
QSGDISP(QMGR)
 END QSTATUS DETAILS
CSQ9022I +MQ11 CSQMDRTC ' DISPLAY QSTATUS' NORMAL COMPLETION
```

MQ Commands cont'd

- DISPLAY CHSTATUS

```
+MQ11 DISPLAY CHSTATUS(MQ12.TO.MQ11) ALL
CSQM293I +MQ11 CSQMDRTC 1 CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I +MQ11 CSQMDRTC  DISPLAY CHSTATUS DETAILS
CHSTATUS(MQ12.TO.MQ11)                                     CHSTATI(21.25.35)
CHLDISP(PRIVATE)                                           CHSTADA(2011-07-30)
CONNAME(::ffff:192.168.1.100)                              BUFSSENT(20)
CURRENT                                                    BUFSRCVD(32)
CHLTYPE(RCVR)                                              MONCHL(HIGH)
STATUS(RUNNING)                                            EXITTIME(0,0)
SUBSTATE(RECEIVE)                                          XBATCHSZ(1,1)
INDOUBT(NO)                                                COMPTIME(0,0)
LSTSEQNO(20)                                               COMPRATE(0,0)
LSTLUWID(AB68344E10000112)                                 STOPREQ(NO)
CURMSGS(0)                                                 KAINT(360)
CURSEQNO(20)                                               QMNAME(MQ11)
CURLUWID(AB68344E10000112)                                 RQMNAME(MQ12)
LSTMSGTI(21.30.14)                                         MCAUSER(MQMTASK)
LSTMSGDA(2011-07-30)                                       LOCLADDR( )
MSGS(20)                                                   BATCHSZ(50)
BYTSSENT(976)                                              MAXMSGL(4194304)
BYTSRCVD(10346)                                            HBINT(300)
BATCHES(18)                                                NPMSPEED(FAST)
 END CHSTATUS DETAILS
CSQ9022I +MQ11 CSQMDRTC ' DISPLAY CHSTATUS' NORMAL COMPLETION
```

MQ Log Data Sets

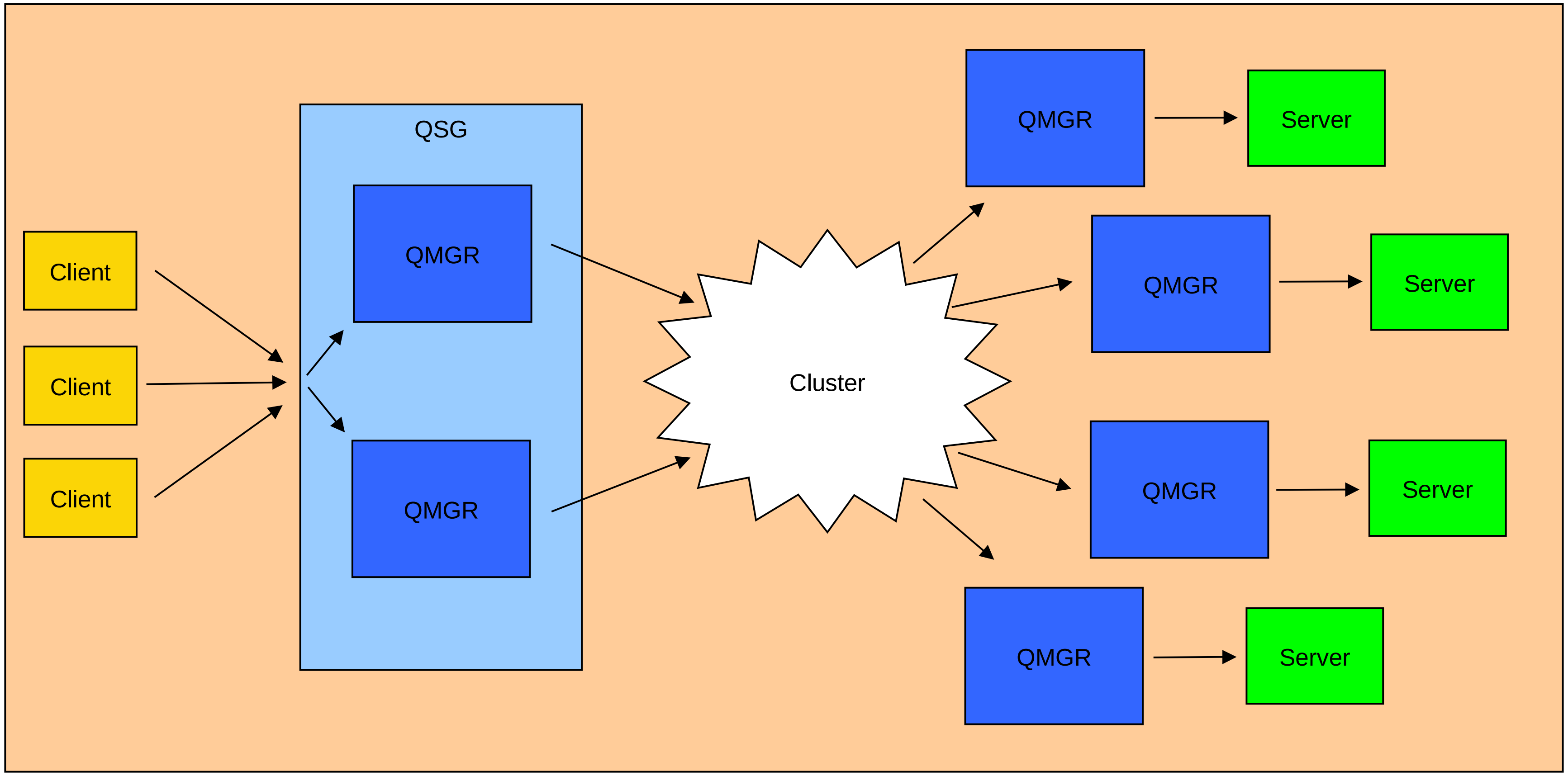
- MQ Log Data Sets record
  - Persistent messages
  - MQ object changes
- CSQ1LOGP utility to format logs
  - EXTRACT function provides a report record for each event
    - Persistent puts and gets
    - Commit and backout
    - Object changes
  - Extracted messages can be replayed to queues

MQ Log Data Sets

- CSQ1LOGP EXTRACT output

```
Time         UR           Userid  App   Job  Data    Queue Name Message   Verb   MD and
             identifier           type       length             key              body

15:08:40.319 00000B2EEE82 JSMITH  BATCH APP1 0155    APP.INPUT  00009101  MQPUT  D4C44040....

15:08:40.319 00000B2EEE82 JSMITH  BATCH APP1 0000                         PHASE1

15:08:40.319 00000B2EEE82 JSMITH  BATCH APP1 0000                         PHASE2

15:08:43.151 00000B2EF3FA DJONES  BATCH APP2 0000    APP.INPUT  00009101  MQGET

15:08:43.151 00000B2EF3FA DJONES  BATCH APP2 0000                         PHASE1

15:08:43.151 00000B2EF3FA DJONES  BATCH APP2 0000                         PHASE2
```

Variable message routes



Identifying message routes

- Activity recording
  - Activity reports generated by applications which perform actions on a message
    - Queue Manager and Chinit
    - User applications
- Can be requested for application messages
- Trace-route messages provide more flexibility
  - dspmqrte tool
    - Generates trace-route requests
    - Collects and displays results

dspmqrte tool

- Test application for submitting trace-route requests and processing responses
- Not available on z/OS, but can connect to z/OS queue manager in client mode

Summary output:

```
C:\>SET MQSERVER=SYSTEM.DEF.SVRCONN/TCP/192.168.1.100(1999)

C:\>dspmqrte -c -q WINQMGR1.APP1.QUEUE -o

AMQ8653: DSPMQRTE command started with options '-c -q WINQMGR1.APP1.QUEUE -o'.
AMQ8659: DSPMQRTE command successfully put a message on queue
'WINQMGR1.APP1.QUEUE', queue manager 'MQ11'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
AMQ8666: Queue 'WINQMGR1.APP1.QUEUE' on queue manager 'MQ11'.
AMQ8666: Queue 'MQ12.TO.WINQMGR1' on queue manager 'MQ12'.
AMQ8666: Queue 'APP1.QUEUE' on queue manager 'WINQMGR1'.
AMQ8652: DSPMQRTE command has finished.
```

dspmqrte tool cont’d

- Detailed output:

```
C:\>dspmqrte -c -q WINQMGR1.APP1.QUEUE –o –v outline

--------------------------------------------------------------------------------
Activity:
 ApplName: 'ebSphere MQ\bin\dspmqrte.exe'
 Operation:
  OperationType: Put
  QMgrName: 'MQ11                                            '
  QName: 'WINQMGR1.APP1.QUEUE                            '
  RemoteQName: 'WINQMGR1.APP1.QUEUE                            '
  RemoteQMgrName: 'MQ12                                            '
--------------------------------------------------------------------------------
Activity:
 ApplName: 'MQ11CHINCSQXRCTL1464FA50    '
 Operation:
  OperationType: Get
  QMgrName: 'MQ11                                            '
  QName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE                  '
  ResolvedQName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE                  '
 Operation:
  OperationType: Send
  QMgrName: 'MQ11                                            '
  RemoteQMgrName: 'MQ12                                            '
  ChannelName: 'TO.MQ12             '
  ChannelType: ClusSdr
  XmitQName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE                  '
--------------------------------------------------------------------------------
```

Delayed messages

- "Missing" messages may just have been delayed
  - Application sees MQRC_NO_MSG_AVAILABLE
  - Message is found on target queue
- Finding processing delays for problem messages
  - CSQ1LOGP
  - Activity reports
- Identifying queue manager components with backlogs
  - Status commands
  - Statistics and accounting data

Real-time monitoring

- DISPLAY QSTATUS

```
+MQ11 DISPLAY QSTATUS(APP1.INPUT) ALL
CSQM293I +MQ11 CSQMDRTC 1 QSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I +MQ11 CSQMDRTC  DISPLAY QSTATUS DETAILS
QSTATUS(APP1.INPUT)
TYPE(QUEUE)
OPPROCS(1)
IPPROCS(0)
CURDEPTH(4)
UNCOM(NO)
MONQ(HIGH)
QTIME(6639576,9403795)
MSGAGE(7)
LPUTDATE(2011-07-30)
LPUTTIME(21.15.57)
LGETDATE(2011-07-30)
LGETTIME(21.16.00)
QSGDISP(QMGR)
 END QSTATUS DETAILS
CSQ9022I +MQ11 CSQMDRTC ' DISPLAY QSTATUS' NORMAL COMPLETION
```

Real-time monitoring cont’d

- DISPLAY CHSTATUS

```
+MQ11 DISPLAY CHSTATUS(MQ11.TO.MQ12) ALL
CSQM293I +MQ11 CSQMDRTC 1 CHSTATUS FOUND MATCHING REQUEST CRITERIA
CSQM201I +MQ11 CSQMDRTC  DISPLAY CHSTATUS DETAILS
CHSTATUS(MQ11.TO.MQ12)                            CHSTATI(09.19.04)
CHLDISP(PRIVATE)                                  CHSTADA(2011-08-04)
XMITQ(MQ11.TO.MQ12)                               BUFSSENT(22)
CONNAME(192.168.1.100)                            BUFSRCVD(13)
CURRENT                                           LONGRTS(999999999)
CHLTYPE(SDR)                                      SHORTRTS(10)
STATUS(RUNNING)                                   MONCHL(HIGH)
SUBSTATE(MQGET)                                   XQTIME(229,167)
INDOUBT(NO)                                       NETTIME(2896,3059)
LSTSEQNO(11)                                      EXITTIME(0,0)
LSTLUWID(C82B9F203F851910)                        XBATCHSZ(1,1)
CURMSGS(0)                                        COMPTIME(0,0)
CURSEQNO(11)                                      COMPRATE(0,0)
CURLUWID(C82B9F21F04E1D5E)                        STOPREQ(NO)
LSTMSGTI(09.21.02)                                KAINT(360)
LSTMSGDA(2011-08-04)                              QMNAME(MQ11)
MSGS(11)                                          RQMNAME(MQ12)
BYTSSENT(6022)                                    LOCLADDR(192.168.1.99(4330))
BYTSRCVD(780)                                     BATCHSZ(50)
BATCHES(11)
 END CHSTATUS DETAILS
CSQ9022I +MQ11 CSQMDRTC ' DISPLAY CHSTATUS' NORMAL COMPLETION
```

Statistics and accounting

- MQ can record statistics and accounting data in SMF
- Performance statistics
  - Record type 115
  - Component related
  - Written at statistics interval
- Accounting data
  - Record type 116
  - Task related
  - Written when task disconnects

How to avoid problems

Detect problems early

- Know what the normal state is for your system
  - MQ joblog messages
  - DISPLAY QSTATUS and CHSTATUS
  - dspmqrte
- Configure instrumentation events
  - Queue manager events
  - Performance events
  - Channel events
  - Configuration events
  - Command events

Know your system

- Queue depths: where are they expected, where are they unusual. Use alerts to get an early warning
- Know commonly issued messages in the joblogs (i.e. certain messages may be issued on a reoccurring basis → know when they may be red herrings)
- Set sensible values on things like max msg size and max queue depth to get an immediate failure rather later performance problems
- Keep reference data
- Trace
- Deal with generated messages (alerts, events, dead letter queue)

Detect problems early cont'd

- System resource monitoring
  - CPU usage
  - I/O
  - Storage
  - Paging
- External monitoring tools
  - Track MQ supplied data (SMF, RMF, events, messages)
  - Show history of data
  - Configure more sophisticated alerts

Thank you for your attention!

- Any questions?