#SHAREorg

SHARE
Technology · Connections · Results

Old and new tracking facility - from Console Tracker to Generic Tracker

Peter Relson
Ulrich Thiemann
IBM

August 13th, 2013
Session 14235

SHARE in Boston

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- The term “tracker” or “tracking facility” is used as short form of “IBM Generic Tracker for z/OS” (new) or “IBM Console Tracking Facility” (old).
- The term Health Checker is used as short form of “IBM Health Checker for z/OS” in this presentation.
- The term “health check” or just “check” is used as short form of “health check for the IBM Health Checker for z/OS” in this presentation.

Agenda

- Objectives
- Overview
- Generic Tracker as new function
- From old to new Tracker
- References
- Summary

Session Objectives

- Introduce the new tracking facility “Generic Tracker”.
- Compare Generic Tracker to the “old” Console Tracking Facility and show how to switch from old to new.

Overview

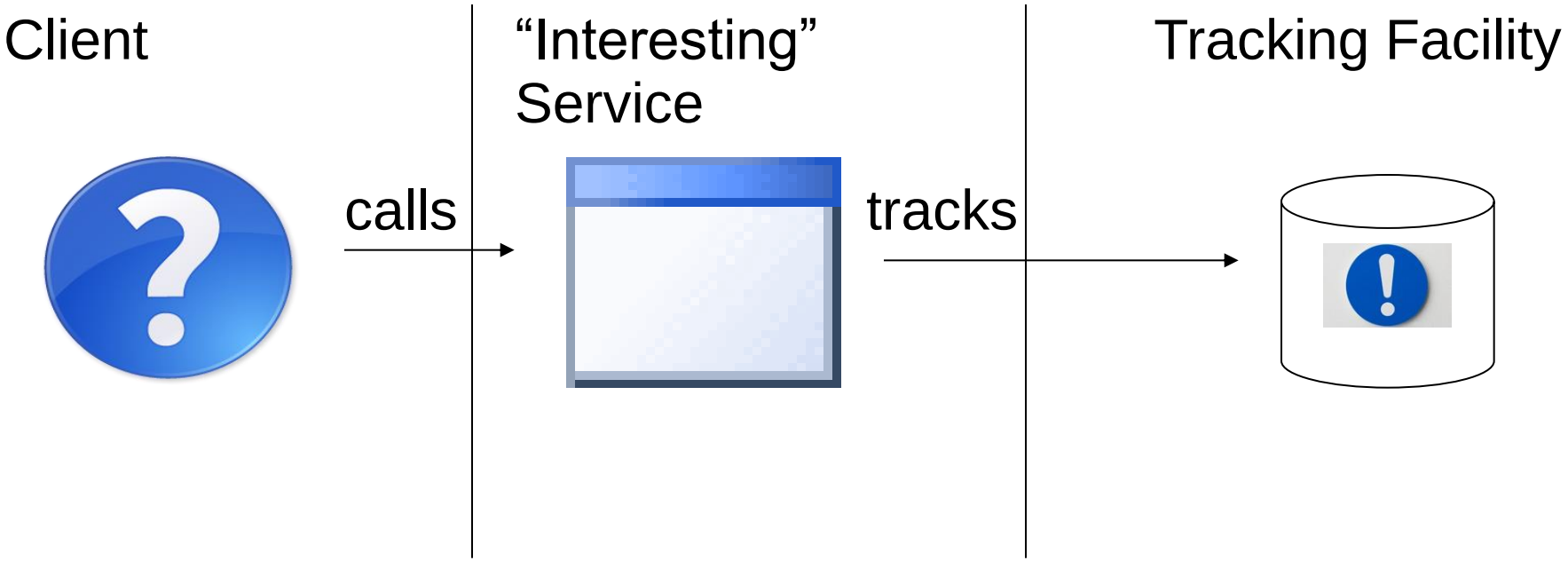
- Problem Statement / Need Addressed
  - The existing “One-byte console ID tracker” started as a single purpose migration aid. Over time other components used it to track events that would aid in their migration efforts. While never designed for such additional use, the Console Tracker reached its limits.
- Solution
  - Provide a “generalized” version of the tracking facility, the new “Generic Tracker”.
- Benefit / Value
  - The new tracking facility allows to track more information and is more flexible when it comes to working with the tracker, making it a general tool to assess migration needs and to assess exploitation of product functions.

Overview – *continued*

- The following describes Generic Tracker as a new function, as if you never heard of the “old” Console Tracker.
- The Migration section later will describe the connection between new and old tracker.

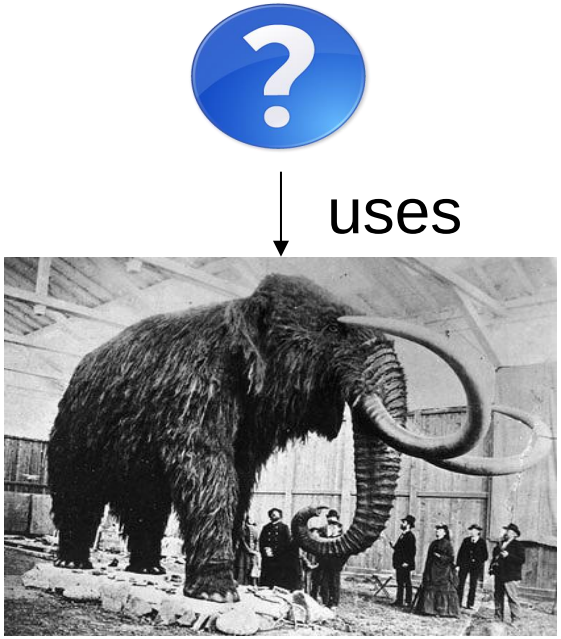
What does it do?

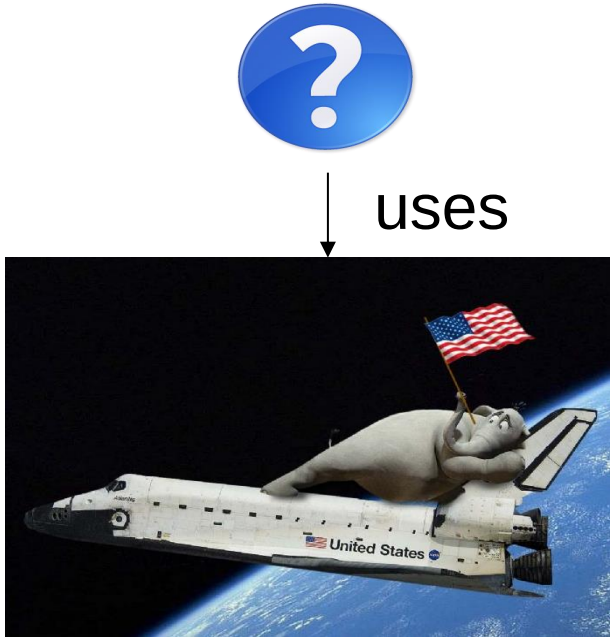
- Finds the name of a calling program and
- Keeps a (“track”) record of the call for you



What to use it for?

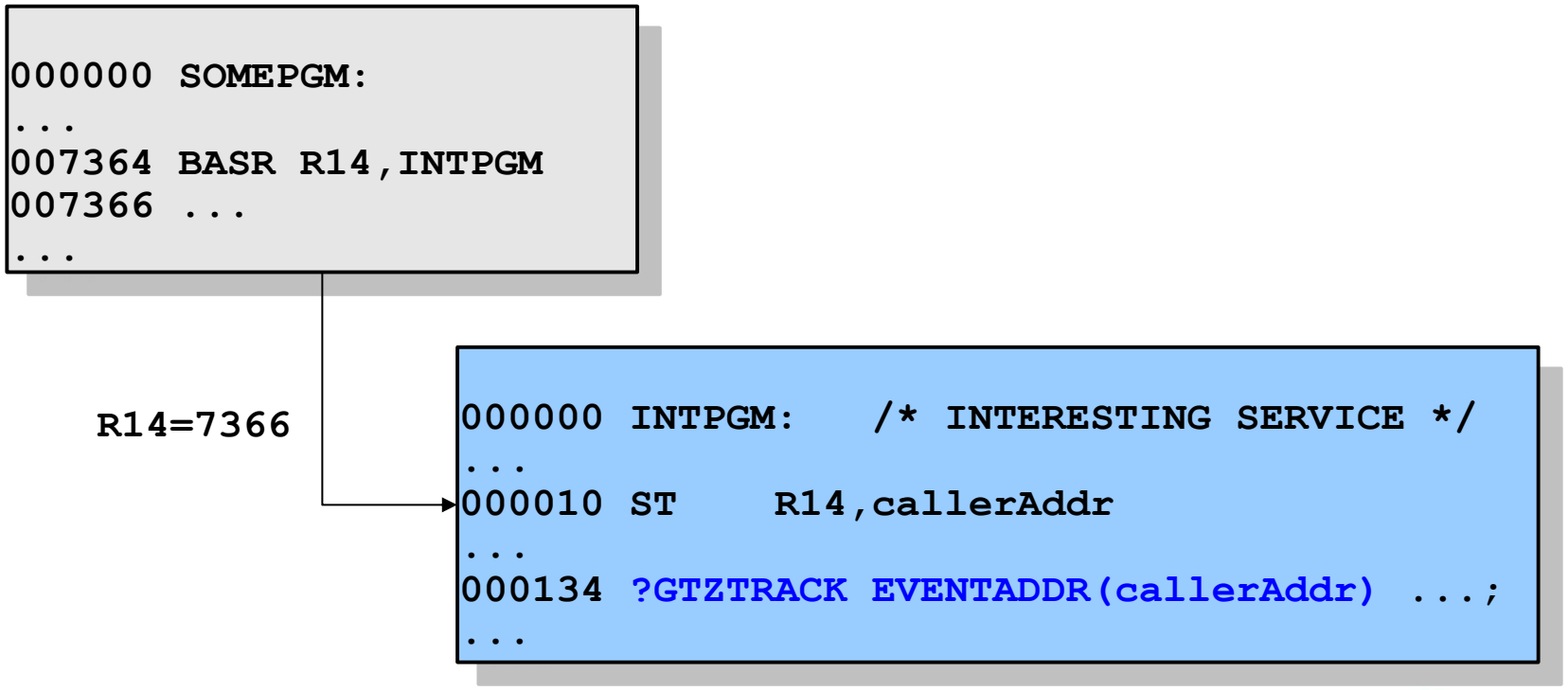
- Migration aid
- Exploitation assessment
- Compliance checking





How to use it?

- As owner of an “interesting” service:
  - Instrument code with GTZTRACK service call



How to use it?

- As system programmer
  - Inspect recorded tracked instances

```
 - SY39  D GTZ,TRACKDATA
GTZ1002I  13.33.59 GTZ TRACKDATA     FRAME  1     F      E   SYS=SY39
FOUND 4 MATCHING TRACKED INSTANCE(S)
INSTANCE 1 ----------------------------------------------------
 EVENTDESC:     'SMS-E:1 DADSM OBTAIN        '
 OWNER:         IBMCNZ                SOURCE:        CNZTRKR
 EVENTDATA:     x0000000000000000     x0000000000C10001
 PROGRAM:       IFCEREP1              PROGRAMOFFSET: x00000000000002C2
 HOMEJOB:       IBMUSER               HOMEASID:      x002C
 EVENTJOB:      IBMUSER               EVENTASID:     x002C
 COUNT:         2                     AUTHORIZED:    YES
 FIRST TIME:    2012-08-21 12:08:21
...
```

What to do with the result?

- Look for clues and report

```
- SY39   D GTZ,TRACKDATA
GTZ1002I  13.33.59 GTZ TRACKDATA     FRAME  1     F      E   SYS=SY39
FOUND 4 MATCHING TRACKED INSTANCE(S)
INSTANCE 1 ------------------------------------------------------
 EVENTDESC:      'SMS-E:1 DADSM OBTAIN          '
 OWNER:          IBMCNZ                SOURCE:        CNZTRKR
 EVENTDATA:      x0000000000000000     x0000000000C10001
 PROGRAM:        IFCEREP1              PROGRAMOFFSET: x00000000000002C2
 HOMEJOB:        IBMUSER               HOMEASID:      x002C
 EVENTJOB:       IBMUSER               EVENTASID:     x002C
 COUNT:          2                     AUTHORIZED:    YES
 FIRST TIME:     2012-08-21 12:08:21
...
```

- OWNER=IBMCNZ and SOURCE=CNZTRKR are special
  - see “From Old to New” migration section later

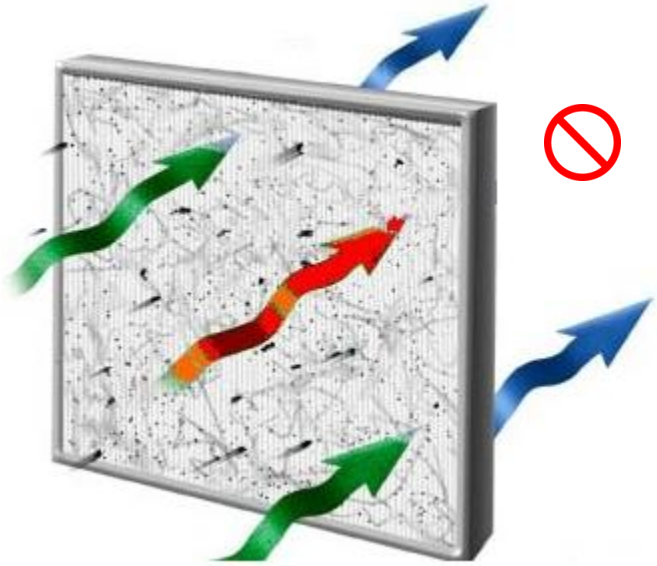
(Intermediate) Summary

- The tracking facility
  - Allows developers (not just IBM) to instrument their code with a “tracking” service
  - Expects the instrumented code to determine the caller-address of this code section
  - Resolves the address to a program name and creates a record of the name and of additional context information in its internal database
  - Allows this information to be retrieved via centralized services for further analysis

That's not all...

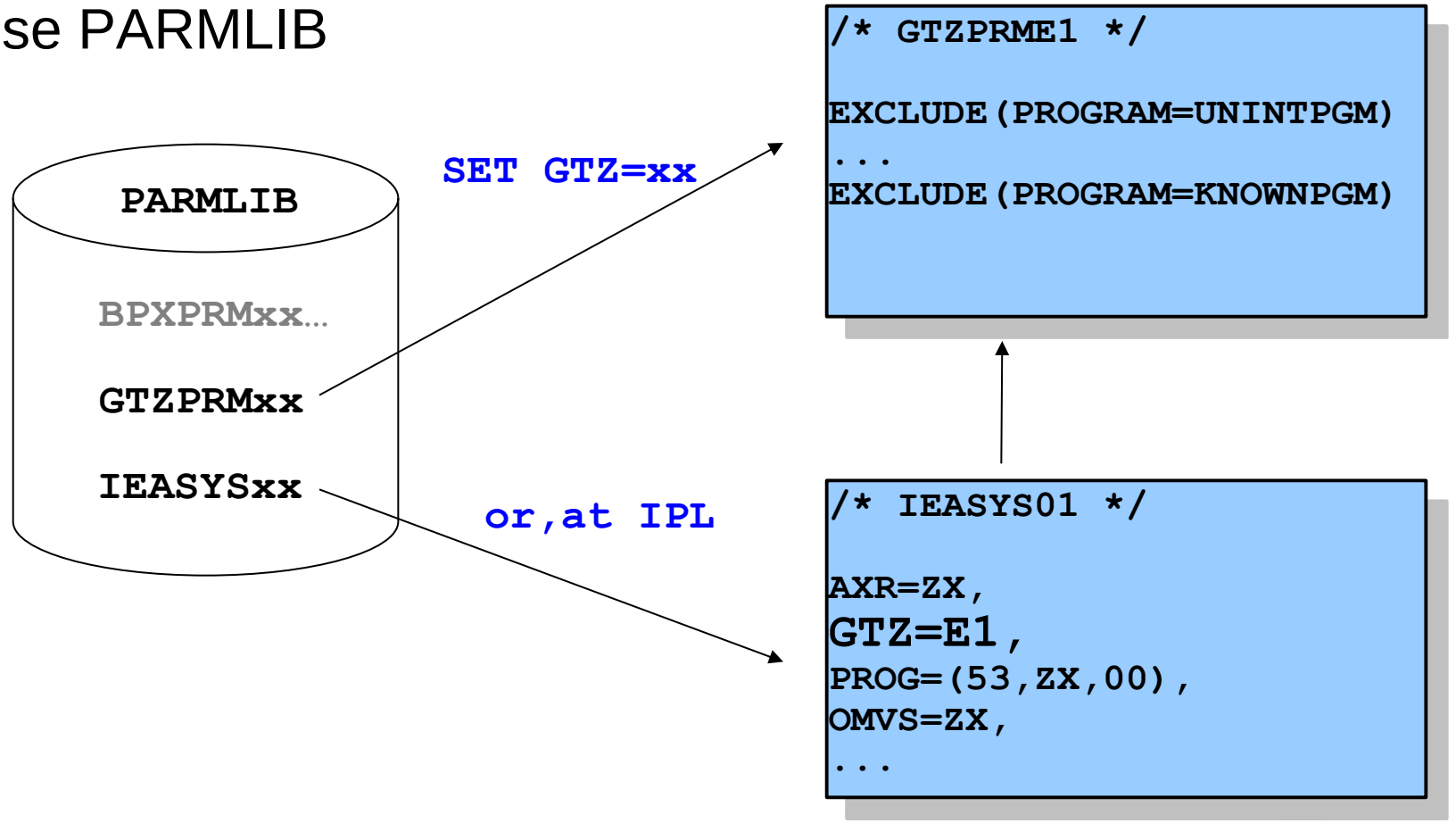
- Can exclude “uninteresting” events from being tracked

```
SETGTZ EXCLUDE(PROGRAM=UNINTPGM)
```



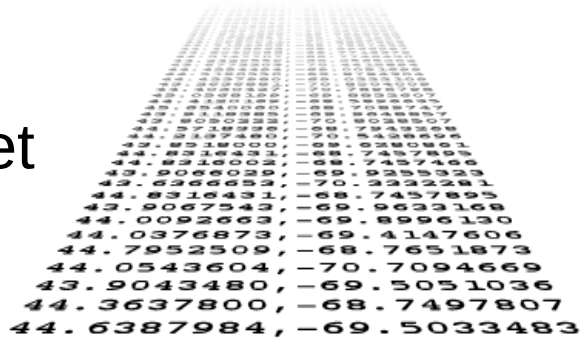
Want to set it and forget it?

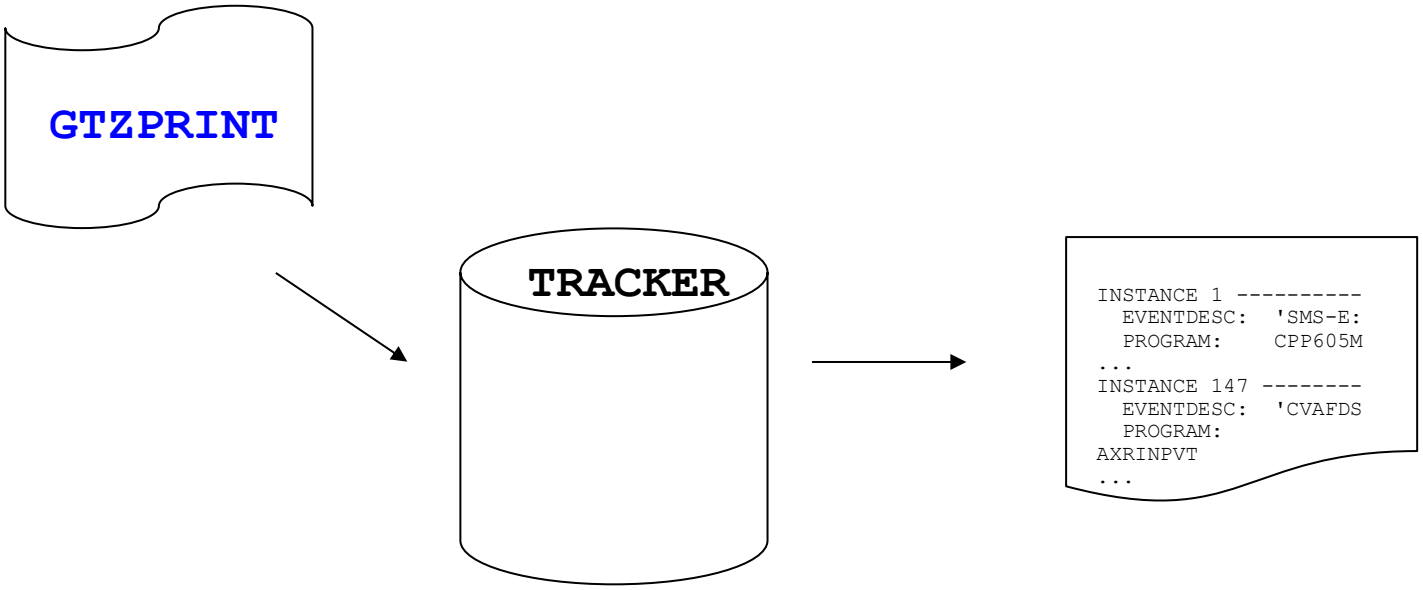
- Use PARMLIB



Too much to look at?

- Store, view, and share it via a dataset





Want to write your own analysis tool?

- Use the GTZQUERY API

Let a health check automate things

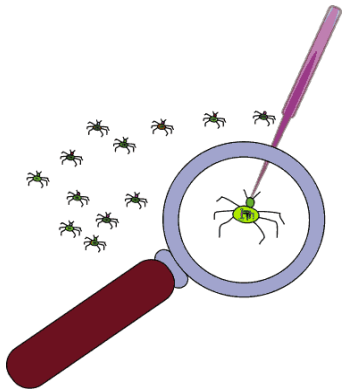
- Use GTZQUERY to write a health check
- Health Checker framework takes care of
  - Automatic scheduling
  - Central reporting

- Check out the GTZSHCK sample health check
  - Can report on specific tracked instances...
  - ...or if *any* instances were found at all
    - see sample prologue for details
  - Use JCL sample GTZSHCKJ to build the sample

For the hard-to-track callers

- Use DEBUG

```
SETGTZ DEBUG(ACTION=ABEND
             REASON=x401
             PROGRAMOFFSET=xB76B2C2
             HOMEJOBNAME=IBMUSER
             SOURCE=CNZTRKR)
```



```
- SY39  D GTZ,TRACKDATA
GTZ1002I  13.33.59 GTZ TRACKDATA      FRAME  1      F       E    SYS=SY39
FOUND 4 MATCHING TRACKED INSTANCE(S)
INSTANCE 1 ----------------------------------------------------
 EVENTDESC:     'SMS-E:1 DADSM OBTAIN        '
 OWNER:         IBMCNZ                SOURCE:         CNZTRKR
 EVENTDATA:     x0000000000000000     x0000000000C10001
 PROGRAM:       *UNKNOWN              PROGRAMOFFSET: x000000000B76B2C2
 HOMEJOB:       IBMUSER               HOMEASID:      x002C
 EVENTJOB:      IBMUSER               EVENTASID:     x002C
 COUNT:         2                     AUTHORIZED:    YES
 FIRST TIME:    2012-08-21 12:08:21
```

Want to start fresh?

- Use CLEAR

```
SETGTZ CLEAR={TRACKDATA|
              EXCLUDE|
              DEBUG|
              ALL}
```

Don't forget

- **Must explicitly enable tracking** (and EXCLUDEs)
  - `SET GTZ=00`
  - **`SETGTZ TRACKING=ON`**

Some interface details – GTZTRACK macro service

- Input
  - OWNER
  - SOURCE or SOURCEPATH
  - EVENTDESC
  - EVENTDATA
  - EVENTASID
  - EVENTADDR
  - EVENTPSW
- Derived output (+ copy of most input)
  - PROGRAM or PROGRAMPATH
  - PROGRAMOFFSET
  - HOMEJOB

Some interface details – GTZQUERY macro service

- GTZQUERY REQUEST(STATUS)
- GTZQUERY REQUEST(EXCLUDE)
- GTZQUERY REQUEST(DEBUG)
- GTZQUERY REQUEST(TRACKDATA)
  [OWNER...]
  [SOURCE/SOURCEPATH...]
  [EVENTDESC...]
  [EVENTDATA...]
  [EVENTJOB...] [EVENTASID...]
  [HOMEJOB...] [HOMEASID...]
  [PROGRAM/PROGRAMPATH...] [PROGRAMOFFSET...]
  - *filter keys allow for wildcards (*, ?)*
- See also output (“answer area”) mapping macro GTZZQRY

Some interface details – DISPLAY GTZ command

- DISPLAY GTZ[,STATUS]
- DISPLAY GTZ,EXCLUDE
- DISPLAY GTZ,DEBUG
- DISPLAY GTZ,TRACKDATA[(*[OWNER=...]…)]*
  - *filter keys allow for wildcards (*, ?)*

- Especially for D GTZ,TRACKDATA see also GTZPRINT

Some interface details – SETGTZ command

- SETGTZ TRACKING={ON|OFF}
- SETGTZ EXCLUDE[=]([OWNER=...]...)
- SETGTZ DEBUG[=](ACTION=...
  REASON=...
  LIMIT=...
  [OWNER=...]...)
- SETGTZ CLEAR={TRACKDATA|EXCLUDE|DEBUG|ALL}

- The EXCLUDE and DEBUG filter keys allow for wildcards (*, ?)

SETGTZ command details – continued

- Can be used as both command and GTZPRMxx parmlib member statement
  - Omit the “SETGTZ” and use in GTZPRMxx
  - Use *key*=*value* or *key*(*value*) syntax for either
    - Traditionally 1st is command, 2nd is PARMLIB syntax
  - Commas (‘,’) are optional as separators and blanks are OK

SETGTZ command details – continued

- Command examples
  - SETGTZ TRACKING=ON
  - SETGTZ EXCLUDE(OWNER=IBM*)
  - SETGTZ DEBUG(ACTION=ABEND,REASON=X’34’,LIMIT=2,OWNER=IBM*)
- GTZPRMxx examples
  - TRACKING(ON)
  - EXCLUDE(OWNER=IBM*)
  - DEBUG(ACTION=ABEND REASON=X’34’ LIMIT=2 OWNER=IBM*)

Some interface details – SET command

- SET GTZ={xx | (xx,...,zz) }
  - Adds configuration statements
- System parameter GTZ specifies initial suffix list
- See also SYS1.PARMLIB(GTZPRM00)
  - Default exclusion list of “known” track events
    - Highly recommended to avoid redundant analysis

Some interface details – GTZPRINT utility

- SYSIN DD accepts one or more PRINT statements
  - PRINT STATUS
  - PRINT EXCLUDE
  - PRINT DEBUG
  - PRINT TRACKDATA[(*[OWNER...] ...)]*
    - filter keys allow for wildcards (*, ?)
  - Syntax borrowed from DISPLAY GTZ command
    - DISPLAY GTZ,STATUS → PRINT STATUS …
- Output routed to SYSOUT DD
- See SYS1.SAMPLIB(GTZPRNTJ)

Some interface details – GTZ procedure

- Used for auto-start
- No need to touch, but has adjustable MEMLIMIT
- See SYS1.PROCLIB(GTZ)

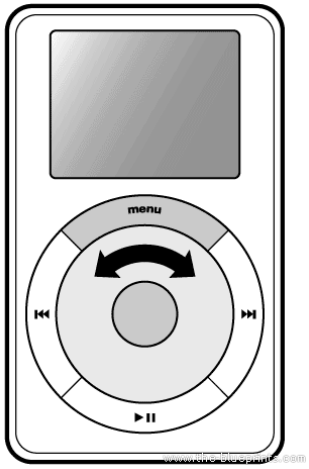
- Needs security setup, if system parameter GTZ is used
  - Startup code needs access to PARMLIB, for example via
    - PERMIT 'SYS1.PARMLIB' CLASS(DATASET) ID(*gtzid*) ACCESS(READ)
    - RDEFINE STARTED GTZ.* STDATA(USER(*gtzid*))

Migration – From old to new Tracker

- More details on the following pages, but in summary
  - Generic Tracker replaces the Console ID Tracking facility (Console Tracker)
  - All commands relating to the Console Tracker have been removed
  - The CNZTRKR macro service continues to be supported and is routed to Generic Tracker under the covers

It's not all new

(Initially) Single Purpose
“One-Byte-Console-ID Tracker”



Generic Tracker

Not without limits, but Generic Tracker...

- Allows more than 1 parmlib member
- Allows more than 1000 unique tracked instances
- Allows and reports full z/OS Unix path names

Adds convenience

- Exclusion lists can be applied “late”
- GTZPRINT replaces log scraping
- More clues about who's tracking what
- Targeted DEBUG option replaces ABEND-all-or-nothing

Adds convenience, for programmers, too

- Parameter list is filled in by macros
- AMODE64 and AR-mode are allowed
- GTZQUERY gives programmatic access to tracker data

From old to new – Macro services

- CNZTRKR still supported, but not recommended.
  - Consider migrating to GTZTRACK
- System converts existing CNZTRKR calls to GTZTRACK at runtime
  - Parameter mapping:
    - TRPL_Track_Info → EVENTDESC
    - TRPL_Track_Value → EVENTDATA
    - TRPL_Violators_Addr → EVENTADDR
    - TRPL_Dont_ABEND → NOABEND
    - ∅ → OWNER='**IBMCNZ**'
    - ∅ → SOURCE='**CNZTRKR**'

From old to new – Commands

- SETCON TRACKING,{ON|OFF} → SETGTZ TRACKING={ON|OFF}
- SETCON TRACKING=ONWITHABEND → SETGTZ DEBUG(ACTION =ABEND...
- DISPLAY OPDATA,TRACKING → DISPLAY GTZ
- SET CNIDTR=xx → SET GTZ=xx
  Conversion tool GTZCNIDT creates GTZPRMxx from CNIDTRxx, see SYS1.SAMPLIB(GTZCNIDJ)

Old commands do not work anymore!

References

- z/OS V2R1 MVS Diagnosis: Tools and Service Aids
  - Overview / Anchor for Generic Tracker
- z/OS V2R1 MVS System Commands
  - SETGTZ, DISPLAY GTZ, SET GTZ
- z/OS V2R1 MVS Assembler Services Reference ABE-HSP
  - GTZTRACK, GTZQUERY – or just use prologues in SYS1.MACLIB
- z/OS V2R1 MVS Initialization and Tuning Guide
  - GTZPRMxx, system parameter GTZ
- http://www.ibm.com/systems/z/os/zos/downloads/
  - Latest IBM supplied GTZPRM00 or just in SYS1.PARMLIB

- Generic Tracker contact: Ulrich Thiemann (thiemanu@us.ibm.com)

Session Summary

- Generic Tracker is a general tool to assess migration needs and to assess exploitation of product functions
  - It provides “tracking” and reporting services

- Generic Tracker replaces the Console ID Tracker
  - Generic Tracker has additional commands, services, options, and tools on top of previously available functionality