IBM Education Assistance for z/OS V2R2

Line item: SMF persistent data & REXX GTZQUERY
Element/Component: BCP Generic Tracker

Material current as of May 2015

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Installation
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Presentation Objectives

- Learn about
  - Track data persistence
  - Query support for REXX

Overview

- Problem Statement / Need Addressed
  - The tracking facility keeps tracking data in dynamic storage and the data has to be explicitly extracted to be saved across IPLs
  - REXX programs can not easily access the reporting functions of existing Assembler service GTZQUERY
- Solution
  - Allow track data to be persisted as SMF records
  - Provide a REXX callable function to query tracking facility information
- Benefit / Value
  - Tracking data can automatically preserved across IPL boundaries and retrieved for later analysis
  - REXX programs, in particular REXX health checks, can report on tracking facility status and data

Overview – Track data, today

- Programs can ask the tracking facility (“Generic Tracker”) to record data via the GTZTRACK service
- If tracking is enabled, the tracking facility stores all unique track data in the dynamic storage of the GTZ address space
- Existing interfaces, like the GTZPRINT utility, or the DISPLAY GTZ operator command, or the GTZQUERY service/API allow the user to extract and store tracking data for analysis

Usage & Invocation – Persisting track data

- All the above will stay unchanged in V2R2
  - The new way to persist TRACKDATA is an *additional* way to store the tracking data, as a “backup”
  - Existing commands will continue to target the “live” data in dynamic storage

- Note: Only TRACKDATA (as recorded from a GTZTRACK call) will be persisted
  - EXCLUDE, DEBUG, or STATUS information is not affected

Usage & Invocation – Persisting track data – Enablement

- The existing operator command SETGTZ supports a new “sub-command” to enable/disable data persistence via SMF
  - SETGTZ PERSIST={**OFF**|SMF}
- By default, data persistence is OFF (disabled)
  - This allows exploiters to plan for additional SMF storage etc. first
- As with existing SETGTZ sub-commands, this new function is also supported as GTZPRMxx parmlib member statement
  - PERSIST({OFF|SMF})
- The enablement status has been added to the output of
  - the existing DISPLAY GTZ[,STATUS] operator command
    - See “PERSIST” field
  - the GTZQUERY REQUEST=STATUS service
  - the new GTZLQRY REXX wrapper for GTZQUERY

Usage & Invocation – Persisting track data – SMF record format

- New SMF record #117 is used to store unique instances of tracking data
  - No “occurrence” count is kept/updated, only the first unique instance of a track record is stored
- Note: Even with PERSIST=SMF, other standard SMF customization statements can prevent the actual recording of track data
- Mapping macro GTZZSMF describes the fields in the new record format
  - These fields are very similar to how track data is returned by the existing GTZQUERY service, see mapping macro GTZZQRY
  - The GTZ mapping macro can also be “included” indirectly via the standard IFASMFR and underlying IFASMFRC SMF record mapping macros

Usage & Invocation – Persisting track data – Retrieve SMF records

- SMF already provides “dump” programs to retrieve “raw”, binary data from different SMF backend stores for SMF records:
  - Program IFASMFDP for SMF records stored in data sets and
  - Program IFASMFDL for SMF data stored in logstreams.
- But, these existing dump programs also support the use of user exit routines to filter/process individual SMF records from the overall set of SMF records stored in the targeted SMF record store (data set or logstream).
- Generic Tracker now provides exit routines for two of those dump program user exits in order to extract and format, **in text form and very similar to the existing DISPLAY GTZ,TRACKDATA output format**, any GTZ SMF records contained in the SMF record source

Usage & Invocation – Persisting track data – Retrieve SMF records...

- SAMPLIB(GTZSMFJ) contains an example invocation and some more details in its job prologue

```
//STEP1 EXEC PGM=IFASMFDP
//INDD1 DD DSN=SYS1.MANA,DISP=SHR
//OUTDD1 DD DUMMY
//SYSPRINT DD SYSOUT=A
//GTZOUT   DD SYSOUT=A,DCB=(LRECL=80)
//GTZPRINT DD SYSOUT=A,DCB=(LRECL=80)
//SYSIN DD *
INDD(INDD1,OPTIONS(DUMP))
OUTDD(OUTDD1,TYPE(117))
USER2(GTZSMFU2)
USER3(GTZSMFU3)
```

Usage & Invocation – Persisting track data – Retrieve SMF records...

- Program GTZSMFU2, to be used as exit routine for exit USER2, which "*is given control only when the SMF data set dump program selects a record to be written*"
  - So GTZSMFU2 gets control for each record selected by any optional general filtering specified via the existing supported input to the SMF dump programs.
  - GTZSMFU2 will ignore any non-GTZ SMF records and will format GTZ SMF records as text in a new output DD, by the name of GTZOUT, of the SMF dump program.
- Program GTZSMFU3, to be used as exit routine for exit USER3, which "*is given control after the output data set is closed*".
  - This is meant to be paired with USER2 exit routine GTZSMFU2 so that GTZSMFU3 can close any GTZ output data sets which GTZSMFU2 opened initially and shared across multiple calls for different SMF records fed into GTZSMFU2.

Usage & Invocation – Persisting track data – Retrieve SMF records...

- On exit the GTZOUT DD contains text like the following example:

```
/* GENERATED BY GTZSMFU2 (HBB77AZ-14260)  2014-09-17 19:40:09 */
-------------------------------------------------------------------------
 INSTANCE:      1                   COUNT:          1
 EVENTDESC:     'GTZFCT neutral event description'
 OWNER:         IBM_GTZ_FCT         SOURCE:         MYSOURCE
 EVENTDATA:     xC7E3E9C6C3E3C2C5   xC5C6F1F500000004
 PROGRAM:       *UNKNOWN            PROGRAMOFFSET: x0000000000000064
 HOMEJOB:       MAINASID            HOMEASID:       x003A
 EVENTJOB:      MAINASID            EVENTASID:      x003A
 AUTHORIZED:    YES                 FIRST TIME:     2014-09-17 18:38:04
 SYSPLEX:       PLEX1               SYSTEM:         SY39
 SMF SID:       SY39
-------------------------------------------------------------------------
 INSTANCE:      2                   COUNT:          1
 EVENTDESC:     'GTZTRACK DIAGNOSE'
 OWNER:         IBMGTZ              SOURCE:         GTZKCTRL
 EVENTDATA:     x0000000000000002   x4040404040404040
 PROGRAM:       GTZKPVT             PROGRAMOFFSET: x000000000002A762
 HOMEJOB:       *MASTER*            HOMEASID:       x0001
 EVENTJOB:      *MASTER*            EVENTASID:      x0001
 AUTHORIZED:    YES                 FIRST TIME:     2014-09-17 19:10:09
 SYSPLEX:       PLEX1               SYSTEM:         SY39
 SMF SID:       SY39

/* GTZSMFU2 END TIME 2014-09-17 19:40:10 */
```

Usage & Invocation – REXX Query interface

- New program GTZLQRY acts as a REXX callable function and is a wrapper for the existing GTZQUERY Assembler service
- All REQUEST types of GTZQUERY are support by GTZLQRY via input variable GTZLQRY_REQUEST:
  - GTZLQRY_REQUEST = “STATUS”
  - GTZLQRY_REQUEST = “TRACKDATA”
  - GTZLQRY_REQUEST = “EXCLUDE”
  - GTZLQRY_REQUEST = “DEBUG”
- The requested output data will be provided as REXX STEM variables
- A sample use of GTZLQRY can be found in the port of existing sample health check GTZSHCK to REXX, now available as SAMPLIB(GTZSHCK**X**)

Usage & Invocation – REXX Query interface – STATUS

- Calling GTZLQRY after setting GTZLQRY_REQUEST = “STATUS” will set a STEM variable GTZQUAAS. (“query answer area for status”) with the following tracking facility status related components:
  - GtzQuaaS.TrackEnabled
  - GtzQuaaS.Full
  - GtzQuaaS.ExcludeNoPrm
  - GtzQuaaS.DebugNoPrm
  - GtzQuaaS.GtzPrmFull
  - GtzQuaaS.ClearedALL
  - GtzQuaaS.ClearedTRACKDATA
  - GtzQuaaS.ClearedEXCLUDE
  - GtzQuaaS.ClearedDEBUG
  - GtzQuaaS.PersistSMF
  - GtzQuaaS.EnabledCount
  - GtzQuaaS.TrackDataEntriesAvailable
  - GtzQuaaS.ExcludeEntriesAvailable
  - GtzQuaaS.DebugEntriesAvailable
  - GtzQuaaS.TrackDataEntriesEncountered
  - GtzQuaaS.ExcludeRejectCount
  - GtzQuaaS.DebugActionCount
  - GtzQuaaS.GtzPrmSuffixes.0
  - GtzQuaaS.GtzPrmSuffixes
  - GtzQuaaS.GtzPrmIplSuffixes.0
  - GtzQuaaS.GtzPrmIplSuffixes.<i>
  - GtzQuaaS.MemAvailPercent
  - GtzQuaaS.SystemName
  - GtzQuaaS.EnabledTOD
- These are modeled after the fields in the GTZQUERY output. See GTZZQRY and the GTZ publications for additional descriptions of the GTZQUAAS components.

Usage & Invocation – REXX Query interface – TRACKDATA

- For GTZLQRY_REQUEST = “TRACKDATA” the function GTZLQRY also accepts the following optional input variables to reduce/filter the result set of currently recorded track data:
  - GTZLQRY_OPTION.MAXRESULTS = { **“ALL”** | *maxresults* }
    - requests to return “ALL” available trackdata entries or only up to *maxresults* number of entries or
  - One or more variables starting with GTZLQRY_FILTER., as described on the next page
    - Note that all filter values, except for SOURCETYPE and PROGRAMTYPE also accept values including wildcard characters:
      - For text values an asterisk ('*') will match zero or more characters of any kind and a question mark ('?') will match a single character of any kind.
      - For numeric or binary values, like EVENTDATA or EVENTASID, only a single asterisk ('*') is supported, as a way to explicitly specify the otherwise implied default value of “match all/match any value”.

Usage & Invocation – REXX Query interface – TRACKDATA...

- See the GTZ publications or the GTZQUERY REQUEST=TRACKDATA documentation for more details on the individual components of input variable GTZLQRY_FILTER.
  - GTZLQRY_FILTER.OWNER = “owner”
  - GTZLQRY_FILTER.SOURCETYPE = { “ALL” | “NOPATH” | “PATH” }
  - GTZLQRY_FILTER.SOURCE = “source"
    - requires GTZLQRY_FILTER.SOURCETYPE = “NOPATH” to be specified as well
  - GTZLQRY_FILTER.SOURCEPATH = “sourcepath"
    - requires GTZLQRY_FILTER.SOURCETYPE = “PATH” to be specified as well
  - GTZLQRY_FILTER.EVENTDESC = “eventdesc"
  - GTZLQRY_FILTER.EVENTDATA = {“eventdata" | “*”}
  - GTZLQRY_FILTER.EVENTASID = {“eventASID" | “*”}
  - GTZLQRY_FILTER.EVENTJOB = “eventjob"
  - GTZLQRY_FILTER.PROGRAMTYPE = { “ALL” | “NOPATH” | “PATH” }
  - GTZLQRY_FILTER.PROGRAM = “program"
    - requires GTZLQRY_FILTER.PROGRAMTYPE = “NOPATH” to be specified as well
  - GTZLQRY_FILTER.PROGRAMPATH = “programpath"
    - requires GTZLQRY_FILTER.PROGRAMTYPE = “PATH” to be specified as well
  - GTZLQRY_FILTER.PROGRAMOFFSET = { “programoffset" | “*”}
  - GTZLQRY_FILTER.HOMEASID = { “homeASID" | “*”}
  - GTZLQRY_FILTER.HOMEJOB = “homejob"

Usage & Invocation – REXX Query interface – TRACKDATA...

- The GTZLQRY_REQUEST=”TRACKDATA” output is provided via
  - Variable GTZQUAAT.TrackDataEntriesAvailable
  - Variable GTZQUAAT.TrackDataEntriesProvided
  - And a set of STEM variables GTZQUAAT.<i> (i=1,...,GTZQUAAT.TrackDataEntriesProvided)
- See the GTZ publications or the mapping macro GTZZQRY and the corresponding GTZQUERY output for more details on the individual components of GTZQUAAT.<i>.
  - GtzQuaaT.<i>.isSourcePath
  - GtzQuaaT.<i>.isProgramPath
  - GtzQuaaT.<i>.isAuthorized
  - GtzQuaaT.<i>.FirstTOD
  - GtzQuaaT.<i>.Count
  - GtzQuaaT.<i>.Owner
  - GtzQuaaT.<i>.Source
  - GtzQuaaT.<i>.EventDesc
  - GtzQuaaT.<i>.EventData
  - GtzQuaaT.<i>.EventJob
  - GtzQuaaT.<i>.HomeJob
  - GtzQuaaT.<i>.Program
  - GtzQuaaT.<i>.ProgramOffset
  - GtzQuaaT.<i>.EventASID
  - GtzQuaaT.<i>.HomeASID

Usage & Invocation – REXX Query interface – EXCLUDE

- For GTZLQRY_REQUEST = “EXCLUDE” function GTZLQRY also accepts an optional input variable to limit the result set of currently active EXCLUDE statements
  - Note: This is more for consistency among the REQUEST types which return a list. EXCLUDE lists are typically “small” (<100)
- GTZLQRY_OPTION.MAXRESULTS = { **“ALL”** | *maxresults* }
  - requests to return “ALL” available EXCLUDE statements or only up to *maxresults* number of statements

Usage & Invocation – REXX Query interface – EXCLUDE...

- The GTZLQRY_REQUEST=”EXCLUDE” output is provided via
  - Variable GTZQUAAE.ExcludeEntriesAvailable
  - Variable GTZQUAAE.ExcludeEntriesProvided
  - And a set of STEM variables GTZQUAAE.<i> (i=1,...,GTZQUAAE.ExcludeEntriesProvided)
- See the GTZ publications or the GTZZQRY mapping macro and the corresponding GTZQUERY output for more details on the individual components of GTZQUAAE.<i>.
  - GtzQuaaE.<i>.OriginType
  - GtzQuaaE.<i>.OriginSuffix
  - GtzQuaaE.<i>.Filter.SourceType
  - GtzQuaaE.<i>.Filter.ProgramType
  - GtzQuaaE.<i>.Filter.Owner
  - GtzQuaaE.<i>.Filter.Source
  - GtzQuaaE.<i>.Filter.EventDesc
  - GtzQuaaE.<i>.Filter.EventData
  - GtzQuaaE.<i>.Filter.EventJob
  - GtzQuaaE.<i>.Filter.HomeJob
  - GtzQuaaE.<i>.Filter.Program
  - GtzQuaaE.<i>.Filter.ProgramOffset
  - GtzQuaaE.<i>.Filter.EventASID
  - GtzQuaaE.<i>.Filter.HomeASID

Usage & Invocation – REXX Query interface – DEBUG

- For GTZLQRY_REQUEST = “DEBUG” function GTZLQRY also accepts an optional input variable to limit the result set of currently active DEBUG statements
  - Note: This is more for consistency among the REQUEST types which return a list. DEBUG lists are typically “very small” (<10)
- GTZLQRY_OPTION.MAXRESULTS = { **“ALL”** | *maxresults* }
  - requests to return “ALL” available DEBUG statements or only up to *maxresults* number of statements

Usage & Invocation – REXX Query interface – DEBUG...

- The GTZLQRY_REQUEST=”DEBUG” output is provided via
  - Variable GTZQUAAD.DebugEntriesAvailable
  - Variable GTZQUAAD.DebugEntriesProvided
  - And a set of STEM variables GTZQUAAD.<i> (i=1,...,GTZQUAAD.DebugEntriesProvided)
- See the GTZ publications or the GTZZQRY mapping macro and the corresponding GTZQUERY output for more details on the individual components of GTZQUAAD.<i>.
  - GtzQuaaD.<i>.OriginType
  - GtzQuaaD.<i>.OriginSuffix
  - GtzQuaaD.<i>.Reason
  - GtzQuaaD.<i>.Action
  - GtzQuaaD.<i>.ActionLimit
  - GtzQuaaD.<i>.ActionCount
  - GtzQuaaD.<i>.Filter.Owner
  - GtzQuaaD.<i>.Filter.Source
  - GtzQuaaD.<i>.Filter.EventDesc
  - GtzQuaaD.<i>.Filter.EventData
  - GtzQuaaD.<i>.Filter.EventJob
  - GtzQuaaD.<i>.Filter.HomeJob
  - GtzQuaaD.<i>.Filter.Program
  - GtzQuaaD.<i>.Filter.ProgramOffset
  - GtzQuaaD.<i>.Filter.EventASID
  - GtzQuaaD.<i>.Filter.HomeASID
  - GtzQuaaD.<i>.Filter.Owner
  - GtzQuaaD.<i>.Filter.Source

Installation

- The new functions are shipped with the Generic Tracker (GTZ) portion of the z/OS base (BCP/HBB77A0), no additional install needed
- If planning to actively exploit the recording of TRACKDATA as SMF records, type 117, additional storage requirements should be considered. As a rough estimate:
  - Average record size is 200-300 bytes, but can rise up to 2K-3K, if maximum length z/OS Unix pathnames are encountered (not common).
  - If a proper “exclusion list” (the recommended list of EXCLUDE statements as shipped via parmlib member GTZPRM00) is set up, an average of 1000-10000 GTZ SMF records might be recorded.
- To exploit the programs GTZSMFU2 and GTZSMFU3 they might have to be configured as allowable USER2 and USER3 exit routines for the SMF dump programs. See the documentation of the SMFDPEXIT parameter of the SMFPRMxx parmlib member.

Presentation Summary

- Track data can now optionally be persisted across IPLs via SMF records of type 117
- Tracking facility status and data is now accessible in REXX via function GTZLQRY

Appendix

- z/OS MVS Diagnosis: Tools and Service Aids (GA32-0905)
  - Overview / Anchor for Generic Tracker
- z/OS MVS System Commands (SA38-0666)
  - SETGTZ, DISPLAY GTZ, SET GTZ
- z/OS MVS Assembler Services Reference ABE-HSP (SA23-1369)
  - GTZTRACK, GTZQUERY – or just use prologues in SYS1.MACLIB
- z/OS MVS Initialization and Tuning Guide (SA23-1379)
  - GTZPRMxx, system parameter GTZ
- http://www.ibm.com/systems/z/os/zos/downloads/
  - Latest IBM supplied GTZPRM00 or just in SYS1.PARMLIB