IBM Education Assistance for z/OS V2R2

Item: REXX support for persistent data
Element/Component: BCP Health Checker

Material current as of May 2015

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Migration & Coexistence Considerations
- Installation
- Presentation Summary
- Appendix

Trademarks

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

Presentation Objectives

- Learn about the persistent data support for REXX checks

Overview

- Problem Statement / Need Addressed
  - The Health Checker framework allows persistent data (saved across IPLs) to be maintained for individual health checks. The current Read/Write interfaces are only available as Assembler services.
- Solution
  - Provide REXX callable functions to read and write persistent data.
- Benefit / Value
  - The growing number of REXX checks (“easiest to implement”) can take advantage of a consistent and easy interface for maintaining persistent data.

Overview – Persistent data

- Health Checker maintains a single data set for persistent data
  - Identified by the HZSPDATA DD statement in the HZSPROC procedure used to start the Health Checker address space,
  - Or identified by the HZSPDATA statement in a HZSPRMxx parmlib member (available in z/OS V2R1 and higher)
  - Required format is shown in SAMPLIB(HZSALLCP): FB 4096...
- Existing services HZSPWRIT and HZSPREAD allow write/read
  - HZSPWRIT can only write for the calling check
  - HZSPREAD can read data for own and other checks (when properly authorized)
- Each check has up to four sets of its own record(s) in HZSPDATA
  - Health Checker keeps data for the current and prior IPL,
  - And for each of those IPLs a first and a most recent instance is kept

Usage & Invocation – Writing persistent data in REXX

- New program HZSLPDWR ("**P**ersistent **D**ata **Wr**ite")
  - REXX callable function
  - Wrapper for existing write service HZSPWRIT
- HZSLPDWR parameters are passed via REXX variables
  - Just like existing Health Checker REXX functions, for example HZSLFMSG, which is the REXX callable wrapper for sending check messages
  - Input variables:
    - HZSLPDWR_BUFFER
      - Data to be written, up to 64K characters/bytes
      - Length is taken from implicit REXX variable length
    - HZS_HANDLE (implicit)

Usage & Invocation – Writing persistent data in REXX, continued...

- Output variables:
  - HZSLPDWR_RC, HZSLPDWR_RSN
    - For any non-zero return code the system will also attempt to issue HZSFMSG REQUEST=STOP REASON=ERROR DIAG=diag
    - See Health Checker User's Guide for list of rc, rsn, and diag values
  - HZSLPDRD_SYSTEMDIAG
    - Additional diagnostic information for non-zero return codes.

Usage & Invocation – Writing persistent data in REXX – Example

```
/* Language: REXX */
 CALL HZSLSTRT
 IF HZS_PQE_FUNCTION_CODE="INITRUN" | HZS_PQE_FUNCTION_CODE="RUN"
 THEN
  DO
   X = 17
   Y = "ABC"
   Z = 31.7
   HZSLPDWR_BUFFER = X","Y","Z
   CALL HZSLPDWR
   IF (HZSLPDWR_RC <> 0)
   THEN SAY "HZSLPDWR RSN=" HZSLPDWR_RSN
  END
 CALL HZSLSTOP
 RETURN
```

Usage & Invocation – Reading persistent data in REXX

- New program HZSL**PDRD** ("**P**ersistent **D**ata **R**ea**d**")
  - REXX callable function
  - Wrapper for existing read service HZSPREAD
- HZSLPDRD parameters are passed via REXX variables
- Input/output is very similar to existing HZSPREAD service

Usage & Invocation – Reading persistent data in REXX – Example

```
/* Language: REXX */
 CALL HZSLSTRT
 IF HZS_PQE_FUNCTION_CODE="INITRUN" | ,
    HZS_PQE_FUNCTION_CODE="RUN" THEN
 DO
  HZSLPDRD_CHECKOWNER = HZS_PQE_CHECKOWNER /* our own */
  HZSLPDRD_CHECKNAME  = HZS_PQE_CHECKNAME  /* our own */
  HZSLPDRD_IPL        = "CURRENT"
  HZSLPDRD_INSTANCE   = "MOSTRECENT"
  /* Default: HZSLPDRD_STARTBYTE  = "FIRST_BYTE" */
  /* Default: HZSLPDRD_DATALEN    = "MAX"        */
  CALL HZSLPDRD
  IF (HZSLPDRD_RC <> 0)
  THEN SAY "HZSLPDRD RSN" HZSLPDRD_RSN
  ELSE
   DO
    PARSE VAR HZSLPDRD_BUFFER X","Y","Z
    SAY "X:" X", Y:" Y", Z:" Z", RETIPLTOD:" C2X(HZSLPDRD_RETIPLTOD)
   END
 END
 CALL HZSLSTOP
 RETURN
```

Usage & Invocation – Reading persistent data in REXX, continued...

- Input variables, required:
  - HZSLPDRD_CHECKOWNER="*checkowner*"
  - HZSLPDRD_CHECKNAME="*checkname*"
    - identify the fully qualified name of the check whose persistent data is to be read
  - HZSLPDRD_IPL={"CURRENT" | "PRIOR"}
    - indicates to read data for the current IPL or data from the previous IPL for the identified check in the HZSPDATA dataset
  - HZSLPDRD_INSTANCE={"FIRST" | "MOSTRECENT"}
    - read from the first data group, or read from the most recently stored data group in the HZSPDATA dataset, relative to the identified IPL (see HZSLPDRD_IPL) for the identified check

Usage & Invocation – Reading persistent data in REXX, continued...

- Optional input
  - to support HZSPREAD-like “one-chunk-at-a-time” reading
  - HZSLPDRD_STARTBYTE={“FIRST_BYTE” | “*offset*” }
    - indicates where to start copying the existing data, in the HZSPDATA dataset, from. Offset is zero-based
    - FIRST_BYTE is a special value for offset zero.
    - Will be ignored (and FIRST_BYTE is used instead) if reading for other than own check
  - HZSLPDRD_DATALEN={“MAX” | “*datalen*”}
    - indicates how much data should be, at most, returned, starting at the offset specified by HZSLPDRD_STARTBYTE
    - MAX is a special value for “return all available data” (starting at the STARTBYTE in effect) up to a maximum of 64K bytes.

Usage & Invocation – Reading persistent data in REXX, continued...

- Output variables:
  - HZSLPDRD_BUFFER
    - Requested data as found in the persistent data set, up to the length specified via HZSLPDRD_DATALEN.
  - HZSLPDRD_BYTESAVAIL
    - Total number of persistent data bytes available for the data record. Might be larger than LENGTH(HZSLPDRD_BUFFER), depending on what HZSLPDRD_STARTBYTE and HZSLPDRD_DATALEN values were used.
  - HZSLPDRD_RETIPLTOD
    - IPL TOD of the persistent data, in STCK format
  - HZSLPDRD_RETPTIME
    - Time the persistent data record, as identified on input, was written. This timestamp is in STCK format.

Usage & Invocation – Reading persistent data in REXX, continued...

- Output variables:
  - HZSLPDRD_RC, HZSLPDRD_RSN
    - For any non-zero return code, except two, the system will also attempt to issue HZSFMSG REQUEST=STOP REASON=ERROR DIAG=diag
      - The two exceptions are the “harmless” no match/no data cases bubbling up from HZSPREAD
      - HZSLPDRD exposes those via RC=8, RSN=**nnnn0881** with nnnn one of
        - **082D** (see HzspreadRsn_NoMatch) and
        - **0830** (see HzspreadRsn_DataDoesNotExist)
    - See Health Checker User's Guide for full list of rc, rsn, and diag values
  - HZSLPDRD_SYSTEMDIAG
    - Additional diagnostic information for non-zero return codes.

Migration & Coexistence Considerations

- None
  - In particular the format of the “HZSPDATA” data set used to store the persistent data has **not** changed. The new REXX functions write (and read) “standard” persistent data records

Installation

- None
  - The new functions are shipped with the Health Checker portion of the z/OS base (BCP/HBB77A0)

Presentation Summary

- Health checks written in System REXX can now use a consistent persistent data interface

Appendix

- Related Publications
  - "IBM Health Checker for z/OS User's Guide" (SC23-6843)
    - Guide and Reference
    - Includes all the details for any new function
    - Includes an inventory of IBM supplied health checks
  - "Exploiting the Health Checker for z/OS infrastructure"
    - Health Checker "hands-on" Redpaper 4590