z/OS 2.4 IBM Education Assistance

Solution (Epic) Name: **Small Health Checker Enhancements**

Solution (Epic) Number(s):

Element(s)/Component(s): **BCP Health Checker**

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Migration & Coexistence Considerations
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Session Objectives

- Provide overview of recent z/OS Health Checker enhancements
  1. Allow UPDATE of REXXHLQ for REXX health checks (SysProg)
  2. Allow ENV N/A status without disabling the health check (Check Writer)
  3. Allow REXX access to “General Health Checker” info and statistics via REXX callable function HZS**L**QRY (AppDev/Check Writer)

Overview – REXXHLQ

- Who (Audience)
  - z/OS Systems Programmer
- What (Solution)
  - Allow UPDATE of REXXHLQ for REXX health checks
- Wow (Benefit / Value, Need Addressed)
  - Previously the high-level qualifier (HLQ) related to REXX check input/output data sets was a fixed, shipped value
    - The HLQ can now be customized, to avoid clashes with existing, general HLQ security setup

Usage & Invocation – REXXHLQ Background

- Health checks written REXX
  - can accept input via a REXXIN data set
  - can produce debug output (DEBUG(ON)) via a REXXOUT data set
- In a non-TSO environment (REXXTSO(NO)), the system dictates the following data set naming convention for REXXIN and REXXOUT:
  - <**REXXHLQ**>.<check-exec-name>.**REXXIN**[.E<check-entry-code>]
  - <**REXXHLQ**>.<check-exec-name>.**REXXOUT**[.E<check-entry-code>]
- For IBM provided health checks, the default REXXHLQ is often IBMUSER, which can clash with an installation's security set up

Usage & Invocation – REXXHLQ Update

- Recommended: Customize REXXHLQ via POLICY UPDATE in HZSPRMxx
  - ADDREPLACE POLICY[(policyname)] [STATEMENT(name)] UPDATE
    CHECK(IBMJES,JES_NJE_SECURITY),
    SEVERITY(LOW),
    INTERVAL(06:00),
    EXCEPTINTERVAL(HALF),
    PARM('NJEEXEC(IRRNJECK)'),
    REXXHLQ(MYHLQ)
    DATE('date_of_the_change'),
    REASON('your update reason')
  - This allows for "set it and forget it"
- For testing, or interim changes, can also use operator command
  - MODIFY *hzsproc*,UPDATE,CHECK…,REXXHLQ=MYHLQ…

Overview – ENV N/A

- Who (Audience)
  - Health check writer
- What (Solution)
  - Allow ENV N/A status without disabling the health check
- Wow (Benefit / Value, Need Addressed)
  - Previously, a check would be disabled (no automatic scheduling anymore) when an “Environment not applicable” was signaled by the check
    - Additional ENV N/A status allows check to stay active and be (auto-) re-scheduled

Usage & Invocation – ENV N/A Background

- Besides a common “successful” or “exception” status, a health check routine might detect “Environment not applicable” at run time
  - The “setting” it is supposed to inspect might be associated with a feature that is not installed/activated (yet) etc.
- Current Health Checker framework interfaces allow check to change its status to “ENV N/A”, via HZS(L)FMSG service:
  - `HZSFMSG REQUEST=STOP REASON=ENVNA`
  - Note the “STOP”:
    - Assumes environment is permanently not applicable and “stops”(=disables) the check
    - Only manual user intervention will make the check run and schedulable again

Usage & Invocation – “Enabled” ENV N/A

- To be used when “ENV N/A” might be only temporary/transitory*:
  - `HZSFMSG REQUEST=HZSMSG REASON=ENVNA`
    - `HZSFMSG`
  - For REXX checks
    - ```
      HZSLFMSG_REQUEST='HZSMSG'
      HZSLFMSG_REASON='ENVNA'
      HZSLFMSG_RC = HZSLFMSG()
      ```
- Will keep the check “Active & Enabled” and system will reschedule as if check run was “successful”
  - But, provides user better indication of check status, as “ENV N/A”, since not “really” successful

*see example on next page
      ```

Usage & Invocation – ENV N/A Example

- Check is run shortly after an IPL completed and Health Checker started as one of the first address spaces, while other system functions are still being started
- Check can signal the new ENV N/A and, during a second check iteration, can re-evaluate availability
  - In absence of any other check-specific indicators, could use
    - Time since Health Checker started (HZSQUERY's HzsquaaGTimeSinceStart)
    - Check (run-) count (PQE_Check_Count)
  - …to decide whether check should continue to signal transient
    HZSFMSG REQUEST=**HZSMSG** REASON=ENVNA,
    or to use more permanent
    HZSFMSG REQUEST=**STOP** REASON=ENVNA

Overview – HZSLQRY

- Who (Audience)
  - z/OS Application Developers and health check writers
- What (Solution)
  - Allow access to “GENINFO” about Health Checker via REXX callable function HZSLQRY
- Wow (Benefit / Value, Need Addressed)
  - Previously the this general information was only available via (HLASM) callable service HZSQUERY
    - Now REXX health checks and other products and applications that use REXX callable functions may query this information

Usage & Invocation – HZSLQRY Background

- Existing (HLASM) service HZSQUERY allows to retrieve general Health Checker information as well as details for individual (sets of) health checks, for example:
  - HZSQUERY REQUEST=GENINFO – framework level info
  - HZSQUERY REQUEST=CHECKINFO – individual check(s) info
  - HZSQUERY REQUEST=MSGBUF – content of individual check message buffers
- Most other (HLASM) services already have a REXX callable counterpart, for example: HZSFMSG has HZS**L**FMSG etc.
- The new HZS**L**QRY function provides coverage for at least
  - HZSQUERY REQUEST=GENINFO

Usage & Invocation – HZSLQRY

- Invoke via
  - `HZSLQRY_REQUEST="GENINFO"`
    `HZSLQRY_RC=HZSQLRY()`
  - Or, just
    `HZSLQRY_RC=HZSLQRY("GENINFO")`
- Find output in REXX variables (+HZSQUERY/HZSQUAA counterpart):
  - HZSQUAAPROCNAME for quaaHeader.HzsquaaHProcname
  - HZSQUAASTID for quaaHeader.HzsquaaHSTID
  - HZSTIMESINCESTART for quaaG.HzsquaaGTimeSinceStart
  - HZSLQRY_RC
  - HZSLQRY_RSN
  - …

Usage & Invocation – HZSLQRY

- HZSPARMLIBMEMBERSUFFIXES.0
  for quaaG.HzsquaaGNumParmlibMemberSuffixes
- HZSPARMLIBMEMBERSUFFIXES.<i> for elements in
  quaaG.HzsquaaGNumParmlibMemberSuffixes
- HZSNUMCHECKSNOTDELETED for quaaG.HzsquaaGNumChecksNotDeleted
- HZSNUMCHECKSDELETED for quaaG.HzsquaaGNumChecksDeleted
- HZSNUMCHECKSDELETEPENDING for
  quaaG.HzsquaaGNumChecksDeletePending
- HZSNUMCHECKSELIGIBLE for quaaG.HzsquaaGNumChecksEligible
- HZSNUMCHECKSCURRENTLYRUNNING for
  quaaG.HzsquaaGNumChecksCurrentlyRunning
- HZSNUMCHECKSINELIGIBLE for quaaG.HzsquaaGNumChecksIneligible
- …

Usage & Invocation – HZSLQRY

- HZSPOLICYNAME for quaaG.HzsquaaGPolicyName
- HZSNUMEXCEPTIONSOUTSTANDING for quaaG.HzsquaaGNumExceptionsOutstanding
- HZSNUMEXCEPTIONSSEVNONE for quaaG.HzsquaaGNumExceptionsSevNone
- HZSNUMEXCEPTIONSSEVLOW for quaaG.HzsquaaGNumExceptionsSevLow
- HZSNUMEXCEPTIONSSEVMEDIUM for quaaG.HzsquaaGNumExceptionsSevMedium
- HZSNUMEXCEPTIONSSEVHIGH for quaaG.HzsquaaGNumExceptionsSevHigh
- HZSNUMPDATARECORDS for quaaG.HzsquaaGNumPDataRecords
- HZSNUMIXGWRITES for quaaG.HzsquaaGNumIXGWRITEs
- HZSNUMBYTESIXGWRITE for quaaG.HzsquaaGNumBytesIXGWRITE

Migration & Coexistence Considerations

- Function HZSLQRY is only supported on z/OS V2R4 and up
  - Fence e.g. by inspecting release indicator in CVT (CVTZOS_V2R4)

- REXXHLQ UPDATEs are only supported on z/OS V2R4 and up
  - If HZSPRMxx is shared with down-level systems, need to fence
  - For example via WHEN statements (needs V2R2 w/ SPE OA49807 and up)
    - ```
      WHEN(&SYSOSLVL. >= 'Z1020400') /* V2R4 and higher */
       DO
        ADDREP POLICY UPDATE CHECK(ownername,checkname)
        REXXHLQ(MYHLQ)
        REASON('REXXHLQ') DATE(20190227)
       END
      ```
      ```

Appendix

- “IBM Health Checker for z/OS User's Guide” (SC23-6843)
  - Guide and Reference
  - Includes an inventory of IBM supplied health checks
    - See also list of security related health checks at https://ek-ibmz.mybluemix.net/health (part of https://developer.ibm.com/tv/enterprise-knights-ibm-z/)
- “Exploiting the Health Checker for z/OS infrastructure”
  - Health Checker “hands-on” Redpaper 4590