IBM Education Assistance for z/OS V2R1

Item: Health Checker Autostart and Miscellaneous New Function
Element/Component: BCP Health Checker

Material is current as of June 2013

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Installation
- Migration & Coexistence Considerations
- Interactions & Dependencies
- Miscellaneous Smaller Updates
- 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 new “Autostart” feature of the IBM Health Checker for z/OS and
  - some miscellaneous smaller items
    - HZSPRINT PARM > 100 characters
    - System name in message buffer
    - Refreshed METAL C sample health checks

Overview

- Problem Statement / Need Addressed:
  - For many releases Health Checker has been helping
    - to ensure system configuration best practices,
    - to prevent system outages, and
    - to successfully migrate from one z/OS release to another.
  - Health Checker was not “ON” by default though and many opportunities to prevent system problems have been missed still.
- Solution:
  - In V2R1 some of the manual configuration steps to have a running instance of Health Checker have been automated and a large subset of the Health Checker function will be “ON” by default.
- Benefit / Value:
  - More installations will receive early warnings and the opportunity to prevent serious system problems.

Usage & Invocation

- The Health Checker address space will now be started automatically at IPL time via procedure HZSPROC.
  - If you have not used Health Checker before, be prepared to initially handle a number of check “exceptions” via messages like HZS0001I (low severity), HZS0002E (medium), and HZS0003E (high).
  - Refer to the individual check message and the details in a check's message buffer on how to “fix” those exceptions.
  - Sometimes your installation might follow different “best practices” and you should customize the check behavior via HZSPRMxx parmlib members to avoid future exception messages.
- The Health Checker User's Guide has good details on how to handle those exceptions.

Usage & Invocation – continued...

- A new system parameter “HZS” can be specified as an alternative, and preferred way to identify the HZSPRMxx parmlib members to be used when Health Checker starts.

Usage & Invocation – continued...

- New special values for parameter HZSPRM of procedure HZSPROC allow to select what HZSPRMxx suffix list to use at Health Checker start
  - HZSPRM=SYSPARM, use suffixes specified via system parameter HZS (can be empty).
  - HZSPRM=PREV, use suffixes which were in use by the previous instance of Health Checker (after a restart e.g. to apply service). Also behaves like SYSPARM for the very first start of Health Checker during an IPL. HZSPRM=PREV is the IBM recommended value.
  - HZSPRM=NONE, to explicitly run without any HZSPRMxx members
- As before, HZSPRM={xx|(aa,...,zz)} tells Health Checker to use the specified suffixes and to in particular ignore system parameter HZS.

Installation

- No “real” install: Health Checker is part of the z/OS base (BCP)
- Customization:
  - If you have not used Health Checker in previous releases (otherwise see the Migration section):
    - Make sure procedure HZSPROC as shipped in IBM's PROCLIB is available in your PROCLIB “concatenation”
    - There are a number of additional setup steps which are described, as in previous releases, in the Health Checker User's Guide, but the majority is optional.
    - Some of those steps are “highly recommended” though and a summary is listed in the following...

Installation - continued...

- Update the shipped HZSPROC procedure to specify a persistent dataset which allows health checks to preserve data across IPLs:

```
//HZSPROC  PROC HZSPRM='PREV'
//HZSSTEP  EXEC   PGM=HZSINIT,REGION=0K,TIME=NOLIMIT,
//       PARM='SET PARMLIB=&HZSPRM'
//*HZSPDATA DD   DSN=SYS1.&SYSNAME..HZSPDATA,DISP=OLD
//         PEND
//         EXEC HZSPROC
```

- Compare SYS1.SAMPLIB(HZSALLCP) for the required format

```
//HZSPDATA DD  DSN=SYS1.system_name.HZSPDATA,DISP=(NEW,CATLG),
//         SPACE=(4096,(100,400)),UNIT=SYSDA,
//         DCB=(DSORG=PS,RECFM=FB,LRECL=4096)
```

- If you do not specify this persistent dataset up front, the system will nag you via HZS0013A – “SPECIFY THE NAME OF AN EMPTY HZSPDATA DATA SET”

Installation - continued...

- Associate a user ID with the HZSPROC address space.

```
RDEFINE STARTED HZSPROC.* STDATA(USER(hcid) GROUP(OMVSGRP))
```

- In particular ensure that this user ID
  - Has an OMVS segment with UID(0) or BPX.SUPERUSER permissions.
    - This is required to run health checks which use z/OS Unix System Services (compare message HZS0109E). Other health checks will run OK without this “superuser” authority.

```
ADDUSER hcid OMVS(UID(yy) HOME('/') PROGRAM('/bin/sh'))
NOPASSWORD
ADDGROUP OMVSGRP OMVS(GID(xx))
CONNECT hcid GROUP(OMVSGRP)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(hcid) ACCESS(READ)
```

  - Has access to your persistent dataset and optionally to other resources (see the Health Checker User's Guide)

Installation - continued...

- Put any (optional) Health Checker customization (health check POLICYs, LOGSTREAM connects,...) into HZSPRMxx parmlib members.

- Set system parameter HZS to the list of suffixes of those HZSPRMxx members, e.g. in IEASYSxx or in reply to message IEA101A at IPL time:

```
HZS=(aa,...,zz)
```

- The procedure parameter HZSPRM in procedure HZSPROC by default references this HZS system parameter, via HZSPRM=PREV.

Installation - continued...

- What if I don't want to run Health Checker on my system?
- That might be OK for a virtual / test system which is OK to fail / be re-IPLed often, but otherwise...
  - Reconsider and do not let an initial “rush” of health check exceptions prevent you from taking advantage of this preventative tool!
- If you really want to disable the auto-start, there is a way, but read on first

Installation - continued...

- How to attack the potential initial “wave” of health check exceptions:
  - Health checks can easily be adjusted to your best practices via check parameters. This is “one time only” work, preserved via HZSPRMxx parmlib members.
  - If you have to, mark “incurable” checks INACTIVE via HZSPRMxx, but let the rest continue to keep an eye on the health of your system.
  - Compare the ADD POLICY UPDATE statement as discussed in the Health Checker User's Guide.
  - Fix the rest to prevent future system problem!
- Are the high severity check exception message filling your console?
  - Consider the CONTROL command, for example: K S,DEL=R
  - Lower the visibility and put this into your HZSPRMxx, temporarily:

```
ADDREPLACE POLICY(HCONLY)
  UPDATE CHECK(*,*) WTOTYPE(HARDCOPY)
  REASON=('STOP RED MESSAGES')
  DATE=(20130408)
ACTIVATE POLICY(HCONLY)
```

Installation - continued...

- If you really have to, set system value HZSPROC to a special value:

```
HZSPROC=*NONE
```

- This will prevent the auto-start, while still allowing a manual start later.

Migration & Coexistence Considerations

- If you used Health Checker in previous releases, the following pages have some recommended actions and caveats

Migration & Coexistence Considerations, continued

- Remove any manual Health Checker start commands
- Typically found in COMMNDxx as “START HZSPROC”
- If not removed, such a second start attempt will be rejected and one of the two following warning message will be issued
  - HZS0101I – “...HEALTH CHECKER... IS ALREADY ACTIVE”
    - When the IPL-time instance is already up and running
  - HZS0116I – “...HEALTH CHECKER... START PENDING
    - When the IPL-time instance is still initializing

Migration & Coexistence Considerations, continued

- If you specified the procedure parameter HZSPRM on the actual start command, like “START HZSPROC,HZSPRM='01'”, then you want to:
  - Move the HZSPRM value to the new system parameter HZS and specify it for example in IEASYSxx
  - Update your procedure to specify HZSPRM=PREV (recommended) or HZSPRM=SYSPARM, to let the system know to use the HZS system parameter
- You could also specify the literal HZSPRM value in your procedure, but HZSPRM=PREV in the procedure is recommended.
- See also the HZSPRM discussion later.

Migration & Coexistence Considerations, continued

- If you renamed the procedure used to start Health Checker...
  - Unlikely and not recommended, but possible
  - Standard name is HZSPROC
- Tell the system about the different name via new system parameter "HZSPROC", for example in IEASYSxx:

```
HZSPROC=MYHCPROC
```

- Just renaming it back to HZSPROC might work, but remember that there likely is a user ID associated with the Health Checker address space via the procedure name. Would need to update that association as well:

```
RDEFINE STARTED HZSPROC.* STDATA(USER(hcid) GROUP(hcidgrp))
```

Migration & Coexistence Considerations, continued

- Want to keep your old Health Checker procedure?
  - The “IBM” HZSPROC is now shipped in the IBM PROCLIB, not in the IBM SAMPLIB anymore
  - Do not copy new IBM HZSPROC from IBM PROCLIB during system upgrade otherwise...
    - it might overwrite your existing HZSPROC, or
    - it might “hide” your renamed procedure since the system will look for procedure “HZSPROC” by default (see also previous page)
  - You probably already keep the IBM PROCLIB towards the end of your PROCLIB concatenation, otherwise your HZSPROC might be hidden by the IBM HZSPROC...

Migration & Coexistence Considerations, continued

- Consider updating your existing procedure to take advantage of the new special values for the procedure parameter HZSPRM
  - IBM recommends HZSPRM=PREV in combination with new system value HZS, instead of the previously used literal list of HZSPRMxx suffixes
  - Note that the HZS system value does not have a default (no “00”). The old HZSPROC coded HZSPRM=“00” as default.

Miscellaneous Smaller Updates – HZSPRINT

- HZSPRINT support for parameter strings longer than 100 characters
  - HZSPRINT is the tool to write check message (buffer) content to a dataset and filter the output by certain criteria
  - V1R10 introduced a number of new filter parameters, in particular TIMERANGE, and the total theoretical parameter length grew to over 100 characters (approximately 180).

Miscellaneous Smaller Updates – HZSPRINT, continued

- In V2R1 HZSPRINT will exploit the new JCL PARMDD support and allow up to 256 characters of parameter data to be passed
  - Trailing blanks per input line do not count
  - Future releases might expand support to the ~32K max length allowed by PARMDD
- The SYS1.SAMPLIB(HZSPRINT) JCL has been updated with an example of the new PARMDD syntax:

```
//HZSPRINT EXEC PGM=HZSPRNT,TIME=1440,REGION=0M,PARMDD=SYSIN
//SYSIN DD *,DLM='@@'
CHECK(*,*)
,EXCEPTIONS
@@
```

Miscellaneous Smaller Updates – HZSPRINT, continued

- No rollback is planned for this HZSPRINT enhancement due to the PARMDD dependency, but here are a few tips to max out the 100 characters for PARM in pre-V2R1 releases:
  - Use wildcards in the check name filter parameter, for example
    instead of `(IBMPFA,PFA_ENQUEUE_REQUEST_RATE)`
    use `(*,PFA_E*)`
  - Avoid having trailing blanks wasting precious space, or having to figure out how to continue a long PARM string across JCL cards, by using JCL SET:

```
//HZSPRINT JOB
// SET PARM1='CHECK(IBMASM,ASM_LOCAL_SLOT_USAGE)'
// SET PARM2=',LOGSTREAM(HZS.HEALTH.CHECKER.LOG)'
// SET PARM3=',EXCEPTIONS'
//HZSPRINT EXEC PGM=HZSPRNT,TIME=1440,REGION=0M,
//   PARM='&PARM1.&PARM2.&PARM3.'
//SYSOUT   DD SYSOUT=A,DCB=(LRECL=256)
```

Miscellaneous Smaller Updates – METAL C sample checks

- The METAL C sample health checks introduced in V1R12 have been (partially) refreshed
- See /usr/lpp/bcp/samples/hzs* for the C source and Makefile
- See SYS1.SIEAHDR.H(HZSH*) for the C includes

Miscellaneous Smaller Updates – System name in message buffer

- A health check's message buffer now contains the name of the system the check ran on
- This makes it easier to associate the check output with the right system, for example when viewing check message buffers via the SDSF CK panel when the SDSF multi-system support is enabled.

```
CHECK(IBMCATALOG,CATALOG_RNLS)
SYSPLEX:    PLEX1      SYSTEM: SY39
START TIME: 02/19/2013 12:16:40.224036
CHECK DATE: 20120827  CHECK SEVERITY: LOW

* Low Severity Exception *

IGGHC111E CHECK(IBMCATALOG,CATALOG_RNLS) found that the Catalog/DADSM
resources do not conform to IBM recommendations...
```

- Note that the SDSF display contains the system name as well, but only on the main panel and much further to the right in the standard setup.

Presentation Summary

- The new Health Checker auto-start aims to protect more systems, with minimal exploitation effort, from system failures caused by suboptimal configuration.

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