#SHAREorg

SHARE
Technology · Connections · Results

IBM Health Checker for z/OS – Intro and next steps

Peter Relson
Ulrich Thiemann
IBM

August 13th, 2013
Session 14298

SHARE in Boston

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.

Session Objectives

- Provide an overview of the IBM Health Checker for z/OS
  - To allow you to take full advantage of this valuable tool
  - To enable you to explore writing your own health checks

Agenda

- Part 1
  - Health Checker framework
  - Health checks
  - Health check alerts
  - Health Checker setup
- Part 2
  - Basic check writing
- References

Health Checker in z/OS

- A component of MVS that identifies potential problems before they impact your system's availability or, in worst cases, cause outages.
- Part of the "base" operating system, the BCP
  - Shipped via FMID HBB77x0
  - Component prefix HZS
- With a system address space, HZSPROC, as backend of the provided services

What does it do?

- Inspects active z/OS and sysplex settings and definitions
  - for deviations from best practices
  - for getting close to critical thresholds
  - for recommended and required migration actions
- Informs the system programmer via detailed messages
- Provides suggested actions and additional references
  - Health Checker itself does not modify the system

The Health Checker advantage

- Health Checker "automates" validation of environment
  - A program, not a programmer, checks for deviations

- Analysis of outages showed:
  - Significant number were avoidable
    - For example, bad configurations with single points of failure
  - Configurations that were less than optimal
    - For example, unnecessary performance bottlenecks

- Situation exacerbated by:
  - Complex parallel sysplex configuration requirements
  - Experienced skills are limited
  - Rare failures mean less experience by operations staff

The Health Checker advantage – continued

- Many options for flexibility:
  - Sometimes, default values are best guesses
  - Best practices may not become known until good exposure in many environments
- Best practices are not widely known or implemented:
  - Many sources of best practices
    - Product pubs, WSC Flashes, White Papers, wizards, …
  - Hard to determine applicability
  - May be out of date
  - Just providing documentation has a limited effect

Health Checker vs. health checks

- One "Health Checker" framework
  - backed by system address space, HZSPROC

- Many health checks
  - Framework "plug-ins"
  - Do the actual "checking" (inspection of settings…)
  - Owned by separate/independent components/products
  - Not just from IBM (~200), but from ISVs and users as well

Health Checker framework

- Maintains a list of known/registered health checks
- Schedules and runs those health checks
  - One time or on an interval schedule
- Provides consistent check message interface with console, SYSLOG, message buffer… output

Health Checker framework, continued

- It's a "live" framework:
  - Own address space (started task "HZSPROC")
  - With live state (private storage) and worker tasks
  - Not (just) static services/APIs
- Available as product in z/OS V1R7 and up

Health Checker Sysplex scope

- Health Checker instances run on single systems in the Sysplex
- Only one instance of Health Checker on a single system
- “GLOBAL” checks run on only one system in a Sysplex
  - Avoids running redundant copies of "Sysplex aware" checks

Migration checks

- An important subset of health checks help with migration

- Shipped INACTIVE by default
  - Find and make ACTIVE when getting ready to migrate
    - F HZSPROC,DISPLAY,CHECK(IBM*,*MIG*)
      - *Mostly CHECK(IBM*,ZOSMIG*), but ICSF is special*
    - F HZSPROC,ACTIVATE,CHECK(IBM*,*MIG*)

- Especially for migration checks look for PTFs
  - Tagged via SMP/E FIXCAT IBM.Function.HealthChecker

How to notice Health Checker alerts

- Manually
  - Action messages on the console

```
*SY40 *HZS0003E CHECK(IBMXCF,XCF_CDS_SPOF):
*IXCH0242E One or more couple data sets have a single point of failure.
```

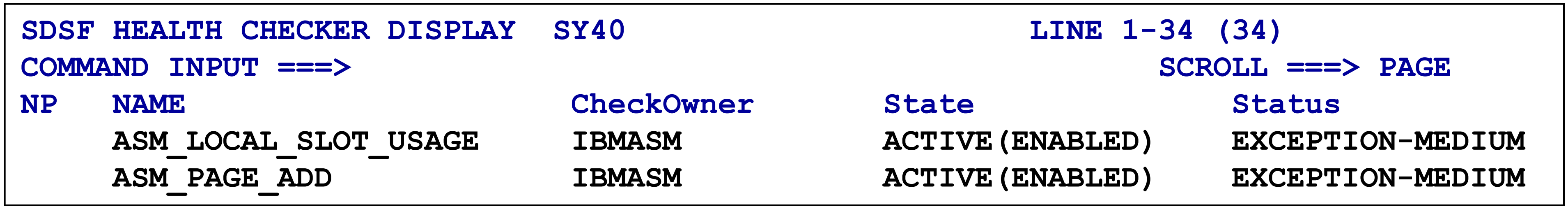
  - “Poll” via command, for example:
    - MODIFY HZSPROC,DISPLAY,CHECKS,EXCEPTION

```
HZS0200I 11.32.59 CHECK SUMMARY      FRAME  1      F      E   SYS=SY40
CHECK OWNER      CHECK NAME                      STATE  STATUS
IBMTSOE          TSOE_USERLOGS                    AE    EXCEPTION-LOW
IBMPDSE          PDSE_SMSPDSE1                    AE    EXCEPTION-LOW
IBMCSV           CSV_LNKLST_SPACE                 AE    EXCEPTION-LOW
IBMCSV           CSV_APF_EXISTS                   AE    EXCEPTION-LOW
```

    - or, HZSPRINT job with option EXCEPTIONS

How to notice Health Checker alerts – Continued

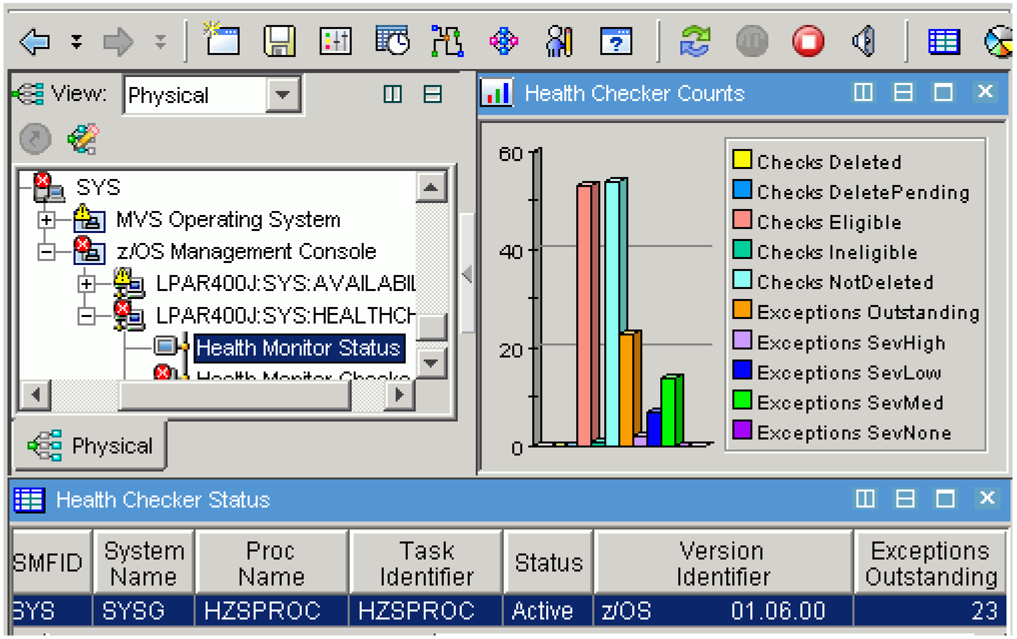
- Manually
  - SDSF CK panel commands, for example: CK E



```
SDSF HEALTH CHECKER DISPLAY  SY40                          LINE 1-34 (34)
COMMAND INPUT ===>                                                SCROLL ===> PAGE
NP   NAME                     CheckOwner        State                Status
     ASM_LOCAL_SLOT_USAGE     IBMASM            ACTIVE(ENABLED)      EXCEPTION-MEDIUM
     ASM_PAGE_ADD             IBMASM            ACTIVE(ENABLED)      EXCEPTION-MEDIUM
```

How to notice Health Checker alerts - continued

- OMEGAMON / Tivoli panels



SMFID	System Name	Proc Name	Task Identifier	Status	Version Identifier	Exceptions Outstanding
SYS	SYSG	HZSPROC	HZSPROC	Active	z/OS 01.06.00	23

How to notice Health Checker alerts - continued

- Automatically
  - Set up notifications via automation products
    - Pager, e-mail, SMS, …
  - Typically based on “generic” HZS message ID in first part of check exception message
    - HZS0003E for HIGH severity exceptions
    - HZS0002E for MEDIUM severity
    - HZS0001I for LOW severity

```
*SY40 *HZS0003E CHECK(IBMXCF,XCF_CDS_SPOF):
*IXCH0242E One or more couple data sets have a single point of failure.
```

What to do with check exceptions

- For each check in exception status:
  - Read the content of the check message buffer

- Message buffer gives all the details needed to “fix”
  - Can be viewed via SDSF CK pane + ‘S’ line command
  - Or via HZSPRINT job output

Check message buffer example

```
CHECK(IBMXCF,XCF_CDS_SPOF)
SYSPLEX:    PLEX1     SYSTEM: SY40
START TIME: 07/19/2013 13:37:29.677274
CHECK DATE: 20070730  CHECK SEVERITY: HIGH
…
IOSPF252I Volumes CPLPKP (0485) and CPLPKA (0487) share the
same physical control unit.
…
* High Severity Exception *

IXCH0242E One or more couple data sets have a single point of failure.

  Explanation:  The couple data set configuration has one or more single
    points of failure.  A failure at one of these points could result in
    loss of a couple data set, system, or even the entire sysplex.
...
 System Programmer Response:  IBM recommends that for maximum
   availability, you operate with both primary and alternate couple
   data set…
   …adding or relocating couple data sets using the SETXCF COUPLE
   command …
```

SHARE in Boston

How to “fix” check exceptions

- “Fix for real” using the information in the message buffer
- Or, “just” adjust a threshold or other check parameter
  - to meet your installation’s “best practices”
  - via F HZSPROC,UPDATE,CHECK(…),PARM… command
    - or SDSF CK panel overtype
  - best made “permanent” by adding UPDATE POLICY statement in a HZSPRMxx parmlib member

Is it fixed?

- System will re-run check automatically after PARM change
  - Can also explicitly request check run
    - SDSF CK panel, ‘R’ line command
    - F HZSPROC,RUN,CHECK(…) command
  - To validate that check reports “success” now

More drastic measures

- Mark individual checks INACTIVE
  - If you really can’t / don’t want to fix
  - Allows all other checks to continue to protect your system
- Most non-applicable checks should already be INACTIVE or DISABLED with ENV N/A

Intermediate remedy

- To help with initial “rush” / high “noise” level on console
  - Don’t forget to make console “scroll”
    - Consider the CONTROL command, for example: K S,DEL=R
  - Lower visibility and put into your HZSPRMxx, temporarily:

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

How to get ready to use Health Checker

- New in z/OS V2.1: “Auto”-start at IPL
  - Before: Put “START HZSPROC” into COMMNDxx…
- Both give you a working Health Checker
  - Additional setup steps are described in the Health Checker User's Guide, but majority is optional.
  - Some steps are highly recommended though and a summary is listed in the following...

Setup – Persistent Data

- Modify HZSPROC to specify a persistent dataset which allows health checks to preserve analysis/comparison 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
```

- Otherwise the system will nag you via
  `HZS0013A-“SPECIFY THE NAME OF AN EMPTY HZSPDATA DATA SET”`

- See 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)
```

Setup – Associated User ID

- Some health checks use z/OS Unix System Services
  - Need an OMVS segment
  - Most health checks will run OK without this
    - But Health Checker will issue warning message HZS0109E
- Some health checks need special authorities
  - To access system resources, including the persistent data dataset
- HZSPRMxx support needs PARMLIB permissions

- Best to associate a user ID with the HZSPROC address space.
  - RDEFINE STARTED HZSPROC.* STDATA(USER(hcid) GROUP(OMVSGRP))

Setup – Associated User ID, continued

- In particular ensure that this user ID
  - Has an OMVS segment with UID(0) or BPX.SUPERUSER permissions.
    - `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 PARMLIB
    - optionally to other resources (see the Health Checker User's Guide)

Setup – Compiled REXX checks

- In particular many migration checks use System REXX
- Comes with extra requirements when compiled REXX is used
  - either the SEAGALT or SEAGLPA library must be in the system search order.
    - SEAGALT is provided in z/OS V1R9 and higher
    - SEAGLPA is provided in the IBM Library for REXX on zSeries product
  - see *IBM Compiler and Library for REXX on System z: User's Guide and Reference*.

Customization – HZSPRMxx

- Put any (optional) Health Checker customization into HZSPRMxx parmlib members
  - health check POLICYs
  - LOGSTREAM connects (for deeper check result history)…
- Set system parameter HZS to list of those HZSPRMxx suffixes
  - HZS=(aa,...,zz) in IEASYSxx – V2.1 only
  - Before V2.1: Update HZSPRM parameter of HZSPROC
- Parameter HZSPRM of procedure HZSPROC by default references this HZS system parameter, via HZSPRM=PREV (V2.1 only)

Other check customization

- PARM string is most common to be updated, but also
  - Check SEVERITY (HIGH, MEDIUM, LOW)
  - Check INTERVAL (scheduling frequency)
  - Check SYNCHVAL
    - schedule more precisely, e.g. to only run during batch hours

Part 2 – Check writing basics

[Real basic information here - More details via session 14232]

- You provide the “inspection” code
  - The “check routine”
- You tell Health Checker where to find it
  - “ADD CHECK”
- Health Checker takes care of the rest
  - Runs check on schedule
  - Reports check messages

Types of checks – “Locale”

- ...in terms of how and where the check routine is provided to Health Checker and finally executed
  - Local checks
    - Check runs in HZSPROC worker task
  - Remote checks
    - Check runs in task of non-HZSPROC, remote address spaces
  - “Hybrid”: System REXX checks
    - Check runs in System REXX (AXR…) address space
- Transparent to users, but important check writer choice
  - For simplicity we will choose REXX in the following

REXX checks

- Check routine is provided as **System REXX exec** in a System REXX library
- Special type of remote check
- Runs authorized
- Can use TSO services

REXX check routine outline

- Establish handshake with Health Checker

- Interpret current check PARM value(s)
- Inspect check specific configuration setting
- Report findings

- Final handshake with Health Checker

Check routine – Handshake with Health Checker

- At check start
  - HZSLSTRT_RC = **HZSLSTRT**()
  - Notifies Health Checker that check routine received control
  - Health Checker provides set of useful HZS* variables
- At check exit
  - HZSLSTOP_RC = **HZSLSTOP**()
  - Let’s Health Checker update status and…
  - …flush and save data used across single check runs

Check routine – Look at parameters

- Use provided HZS* variables to
  - Check for parameter changes (or on first check run)
  - Parse parameter, as needed
  - Store found value(s) for later check runs

```
IF HZS_PQE_LOOKATPARMS = 1 THEN
 DO
   PARSE UPPER VAR
      HZS_PQE_PARMAREA,"LIMIT("Limit_Value")"
   HZS_PQE_CHKWORK = Limit_Value
 END
ELSE Limit_Value = HZS_PQE_CHKWORK
```

Check routine – Inspect settings

- Many REXX services available to inspect storage, system settings, …
- Decide on success or exception

```
IF HZS_PQE_FUNCTION_CODE = "INITRUN" | ,
   HZS_PQE_FUNCTION_CODE = "RUN"
THEN
  DO
   /* Any real checking goes here, for example comparing
      the current LIMIT parameter value against a system
      value. In this sample we just report success every
      other check run and an exception in between... */
   IF (HZS_PQE_CHECK_COUNT // 2) = 1
   THEN /* Report success */
   ELSE /* Report exception */
  END
```

Check routine – Report Success

- “All is good” confirmation

```
HZSLFMSG_REQUEST = "DIRECTMSG"
HZSLFMSG_REASON  = "CHECKINFO"
HZSLFMSG_DIRECTMSG_ID   = "XYZH0001I"
HZSLFMSG_DIRECTMSG_TEXT = "All is well with
  limit xyz"
HZSLFMSG_RC = HZSLFMSG()
```

  - Note the use of “embedded” message text
    - **DIRECTMSG** available since z/OS V1R12

Check routine – Report Exception

- Deviation found, approaching limit, …

```
HZSLFMSG_REQUEST = "DIRECTMSG"
HZSLFMSG_REASON  = "CHECKEXCEPTION"
HZSLFMSG_DIRECTMSG_ID   = "XYZH0002E"
HZSLFMSG_DIRECTMSG_TEXT = "Bad limit xyz."
HZSLFMSG_DIRECTMSG.EXPL = "Limit xys has been reached..."
HZSLFMSG_DIRECTMSG.SYSACT = "The system continues processing."
HZSLFMSG_DIRECTMSG.ORESP = "Report this error to the System Programmer."
HZSLFMSG_DIRECTMSG.SPRESP = "Make more xyz available...."
HZSLFMSG_DIRECTMSG.PROBD = "For problem determination, ...."
HZSLFMSG_DIRECTMSG.SOURCE = "<owning product>"
HZSLFMSG_DIRECTMSG.REFDOC = "Look at the following manuals",
 "to explain the error message further or help diagnose",
 "and correct the problem reported...."

HZSLFMSG_RC = HZSLFMSG()
```

Register the health check

- Have HZSPRMxx parmlib member with ADD

```
ADDREP CHECK(MYPROD,PROD_LIMIT_CHECK)
       EXEC(PRDLIMCK)
       REXXHLQ(IBMUSER)
       MSGTBL(*NONE)
       PARMS('LIMIT(47)')
       SEVERITY(MEDIUM)
       INTERVAL(0:30)
       DATE(20130630)
       REASON('Check PROD LIMIT')
```

← Your REXX exec member in e.g. SYS1. SAXREXEC

- Tell Health Checker about it
  - MODIFY HZSPROC,ADD,PARMLIB=xx

See the results of the check run

- On the console

```
SY39  HZS0002E CHECK(MYPROD,PROD_LIMIT_CHECK):
XYZH0002E Bad limit xyz.
```

- On the SDSF CK panel

```
SDSF HEALTH CHECKER DISPLAY  SY39                      LINE 57-112 (159)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP   NAME                        CheckOwner     State                Status
     IXGLOGR_ENTRYTHRESHOLD      IBMIXGLOGR     INACTIVE(ENABLED)    INACTIVE
     IXGLOGR_STAGINGDSFULL       IBMIXGLOGR     ACTIVE(ENABLED)      SUCCESSFUL
     IXGLOGR_STRUCTUREFULL       IBMIXGLOGR     ACTIVE(ENABLED)      SUCCESSFUL
     JES2_Z11_UPGRADE_CK_JES2    IBMJES2        ACTIVE(ENABLED)      SUCCESSFUL
     OCE_XTIOT_CHECK             IBMOCE         ACTIVE(ENABLED)      EXCEPTION-LOW
     PDSE_SMSPDSE1               IBMPDSE        ACTIVE(ENABLED)      EXCEPTION-LOW
     PROD_LIMIT_CHECK            MYPROD         ACTIVE(ENABLED)      EXCEPTION-MEDIUM
```

See the results of the check run – continued

- In the message buffer
  - Line command ‘S’ (Browse Status) on the SDSF CK panel

```
CHECK(MYPROD,PROD_LIMIT_CHECK)
SYSPLEX:    PLEX1     SYSTEM: SY39
START TIME: 07/20/2013 13:22:59.792932
CHECK DATE: 20130630  CHECK SEVERITY: MEDIUM
CHECK PARM: LIMIT(47)


* Medium Severity Exception *

XYZH0002E Bad limit xyz.

  Explanation:  Limit xys has been reached...
```

SHARE in Boston

More about REXX checks, check writing…

- Check out SYS1.SAMPLIB(HZS*)
  - HZSSXCHN and HZSSXCHK are REXX sample checks

- Session 14232 covers check writing in more depth

References

- SHARE Boston 2013 – Session 14232
  - “IBM Health Checker for z/OS
    - V2R1 Updates
    - Check writing details and comparisons”

- “IBM Health Checker for z/OS User's Guide” (SC23-6843)
  - Guide and Reference
  - Includes an inventory of IBM supplied health checks

- “Exploiting the Health Checker for z/OS infrastructure”
  - Health Checker “hands-on” Redpaper 4590

- Health Checker framework contact and to direct questions about individual health checks:
  - Ulrich Thiemann (thiemanu@us.ibm.com)