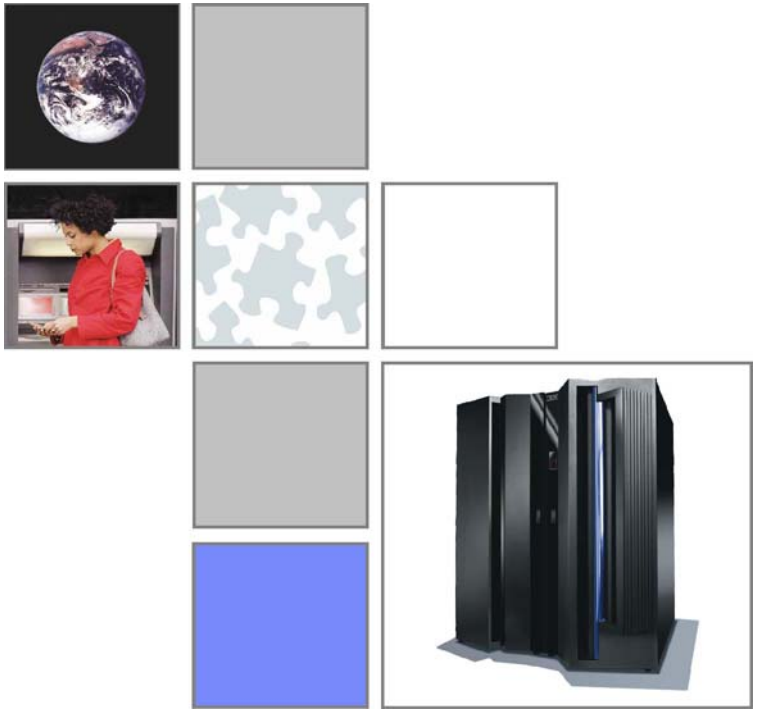
IBM Washington Systems Center

IBM Health Checker for z/OS



Riaz Ahmad
IBM Washington Systems Center
Gaithersburg, Maryland

ON DEMAND BUSINESS™

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

APPN*
DB2*
e-business logo*
Enterprise Storage Systems
ESCON*
FICON
GDPS
Geographically Dispersed Parallel Sysplex
HiperSockets
IBM*
IBM logo*
IMS
Magstar*
MVS
Netfinity*
OS/390*
Parallel Sysplex*
PR/SM
S/390*
S/390 Parallel Enterprise Server
Virtual Image Facility
VM/ESA*
VSE/ESA
VTAM*
WebSphere
z/Architecture
z/OS
z/VM
zSeries

IBM @server zSeries

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
LINUX is a registered trademark of Linus Torvalds
Penguin (Tux) complements of Larry Ewing
Tivoli is a trademark of Tivoli Systems Inc.
Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

IBM considers a product "Year 2000 ready" if the product, when used in accordance with its associated documentation, is capable of correctly processing, providing and/or receiving date data within and between the 20th and 21st centuries, provided that all products (for example, hardware, software and firmware) used with the product properly exchange accurate date data with it. Any statements concerning the Year 2000 readiness of any IBM products contained in this presentation are Year 2000 Readiness Disclosures, subject to the Year 2000 Information and Readiness Disclosure Act of 1998.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Agenda

- Why Healthchecker
- Some Recent History
- Healthchecker overview and installation
- Health Checker restructure in z/OS 1.7

What is the problem ?

- In depth analysis of outages show:
  - A significant number were avoidable
    - *bad* configurations
      - Single points of failure
  - Non-optimum configurations
    - Stressing key sysplex SW in unique ways
    - Unnecessary performance bottlenecks

What is the problem?

- Parallel Sysplex design allows for elimination of single points of failure, but:
  - Complex configuration requirements
  - Skills are at a premium
  - Speed of recovery operations are critical
- Failures are rare
  - Operations and System Programmers caught off guard
- Sympathy Sickness can occur

What is the problem?

- Best practices are not widely known and implemented|
  - Many sources of best practices materials
  - Multiple product publications, Redbooks, WSC Flashes & White Papers, Wizards, etc.
    - Voluminous, generic
      - Difficult to determine applicability
    - Static, point in time
    - Overwhelming
  - Documentation has limited affect

What is the problem?

- Designers and developers do not know what they do not know
  - Multiple tuning knobs for flexibility
  - Sometimes, default values are best guess
  - Some best practices not known until real customer experiences from multiple production environments
- Need the ability to improve availability characteristics

What is the Objective of the Healthchecker?

- Provide a way to more easily and effectively assist installations to implement best practices
  - Proactive scan and identification of exceptions
- Environment Sniffer
  - Programmatically check various settings on the system
    - In storage checks - not a PARMLIB scan
    - Check against known best practices list
    - Notify when exceptions are found

What is the Objective of the Healthchecker?

- Spread lessons learned from
  - Multiple environments
  - Installations
  - internal experiences

- This all boils down to:
  - ***Outage avoidance***

Proof of Concept

- Developed a prototype that included a set of checks
  - Many checks from Parallel Sysplex Availability Checklist
  - From WSC experience of doing sysplex availability studies
- Run a batch job, get a report
- Added WTO of exceptions based on customer input

Proof of Concept

- Made available via web download
  - Large number of downloads
    - Customer interest shown
- Have learned many things:
  - Need ability to override supplied best practice value
    - But with strict controls
  - Need to expand the scope of components doing checks

Proof of Concept

- Made available via web download
  - Large number of downloads
  - Three updates to prototype
  - Active customer and IBMers defining new requirements
- Available on z/OS web site:
  - http://www.ibm.com/servers/eserver/zseries/zos/downloads/

Existing Health Checker

- Free “as-is” tool that can be downloaded from the Web
  - Upload to OS/390 R10 or z/OS system
  - Run as a batch job
  - View exception messages and reports
  - Make suggested changes manually
- Mostly a **Configuration Checker**
- Not an ***msys* for Operations** replacement
  - No automatic correction of problems
  - No online panel displays

Customers have asked for

- A more formal product
  - With formal support
- Checks from more z/OS components
- Checks from more IBM products
- Checks from ISV products
- Ability to write their own checks

IBM z/OS V1.6 Announcement (204-180)

- IBM Health Checker for z/OS will be a new base function in z/OS 1.7 (FMID HZS7720)
  - Checks delivered separately from the framework, can be added dynamically
  - Checks delivered by elements and components as PTFs
  - User overrides check defaults via HZSPRMxx parmlib updates or MODIFY command
- The Framework and most checks intended to be made available as z/OS web download for z/OS releases V1.4, V1.5, and V1.6
- Initial support for most existing checks with plans for incremental delivery of new checks
- SDSF support for managing checks with CK panel

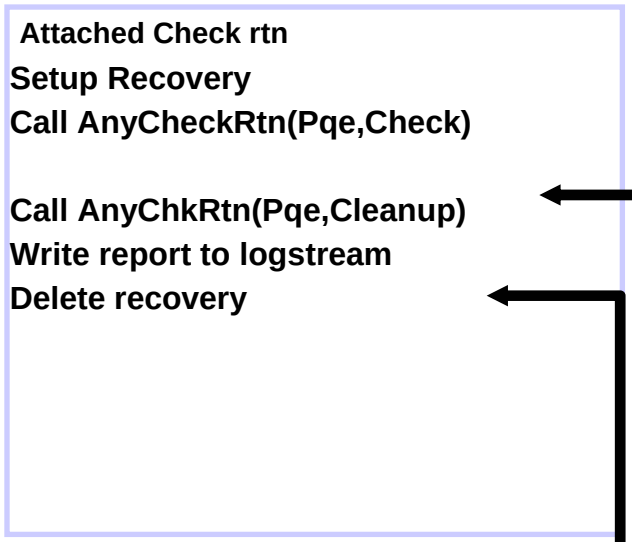
Health Checker Restructure

- Moving from self contained batch job to Started task providing services
  - Allow checks to be added dynamically
    - No previous knowledge of check required by the HC backbone
  - Log results to MVS Logger logstream
  - Provide check management services
  - Long running STC with ability to re-execute checks on interval basis
    - Intervals are unique to each check
    - Interval values from 1 minute to 43 days

New Structure Overview

- Each check has 3 parts
  - The dynamic exit routine that identifies the check to the Healthchecker
  - The check itself
  - A message table to define messages that are issued by the check

Check Runs in Health Checker Subtask



The check is attached by the JOBSTEP TCB for execution

Recovery;
Call AnyChkRtn(Pqe,Cleanup)
Write report to logstream

- **AnyChkRt:**
  - **Function(Null)?**
    - **Return**
  - **Funtion(Init)?**
    - **.....**
  - **Funtion(Delete)?**
    - **.....**
  - **Funtion(Check)?**
    - **EntryCode(AnyChk1)?**
    - **Obtain storage for this iteration of AnyChk1**
    - **Execute**
  - **Funtion(Cleanup)?**
    - **EntryCode(AnyChk1)?**
    - **Release storage for this iteration of AnyChk1**

Checks

- Checks run in the Health Checker address space
- Checks are given a 2K persistent area to save data between check iterations
- 2K Check status area (Current overrides)
- Supervisor, Key 8
- Messages via HZSFMSG service
- Checks are independent of the IBM Health Checker for z/OS component.
  - Check will be shipped by individual component.
    - Expect 3rd party checks to be written
  - SDSF support to modify checks, and view output
    - Expect additional IBM and 3rd party add-ons in the future

Dynamic Exits To Add Checks

- Each dynamic exit routine can either be added via operator command or via an API
- Any dynamic exit routine that is added prior to the start of the Healthchecker, will be invoked when Healthchecker is started
- Healthchecker must be told to run to pick up new checks that are added after Health Checker is started.
- The dynamic exit routine uses the HZSADDCK macro to define one or more checks

Check Structure

- 32 bytes check name and 16 bytes check owner (Company name, and component)
- Entry code (used by the check routine when a single check routine has multiple functions)
- The date the *best practice* values were recommended
- 126 bytes reason that summarizes why the check was written
- The severity of the problem(s) the check is looking for.
- Any default parameter values.
- The default Interval
- The name of the check load module
- The name of the check message table

Check Structure

- Each check is called with a check entry code as defined by HSADDCHK
- Function code:
  - Initialization - Initialization processing (once per life of check)
  - Verify installation parameters
  - Any processing that should be done one for the life of the check
  - Check - Normal check processing
  - Check_cleanup - free any storage obtained during the check.
  - Check_delete - cleanup for any processing done during check initialization

Check Message Table

- Each check has a message table
- Common look and feel
- Structured diagnostic message
- Each message is owned by the check.
- Exception messages contain the WTO text
- Message language based on XML/SGML
- Message source is converted to an assembler file that must be compiled and linked to create the message load module that is included with the check.

Messages

- Checks issue both verbose (‘configuration is good' messages) and exception messages.
- Check output
  - WTOs – exception messages are written as a HSZ (Healthchecker) message number and the component message ID follows HZS msg: **HZS001I IXL002I . . .**
    - Output: All messages are written to the REPORT file (last instance of check)
    - Check history via MVS Logger logstream
    - When an exception message is written, a summary WTO is written to outline the problem

Installation

- Allocate HZSPDATA data set
  - To save checks data between restart
- Set up HZSPRINT utility
  - The Health Checker retains only the check results from the last iteration
    - Kept in a message buffer
    - For historical data define Logger logstream
- Security definitions
  - Users looking at check output require access to resources
    - Multilevel security
- Create HZSPRMxx PARMLIB member
- Copy HZSPROC to PROCLIB

External Interfaces

- Parmlib Support HZSPRMxx
  - Concatenation of members supported
  - Cross Component support
  - User overrides to:
    - Severity, WTO descriptor codes, intervals, active or inactive, categories, parameter values
- Categories
  - Installations can group multiple checks
    - Perform actions against categories
    - One check can be in up to 16 categories
- Operator Interfaces
  - Command interface
    - Display Command
    - Modify Checks
      - Run now, pause, refresh, etc.
  - SDSF CK panel

Operator Commands

- DISPLAY (STATUS/CHECKS/POLICY/DELETED)
- SET/ADD PARMLIB
- LOGGER (ON|OFF)
- ADDNEW
- DELETE target
- ACTTIVATE/DEACTIVATE target
- REFRESH target
- RUN target
- ADDNEW/ADDREPLACE POLICY ….
- UPDATE target update_values
- STOP

SDSF CK Panel

- Display checks, attributes, and status, taking advantage of standard SDSF sort, filter, and arrange support
- Alter check attributes
  - status, interval, severity, category, and WTO descriptor
- Browse check output for the most recent check
- Print check output or sent it to a data set

SDSF: Sample CK Display

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HEALTH CHECKER DISPLAY  SYSB                        LINE 10-33 (33)
COMMAND INPUT ===>                                          SCROLL ===> CSR
PREFIX=LOG*  DEST=(ALL)  OWNER=*  SYSNAME=*
NP   NAME                            State              Status             SysN
     CNZ_SYSCONS_ROUTCODE            ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     GRS_CONVERT_RESERVES            ACTIVE(ENABLED)    EXCEPTION-LOW      SYSB
     GRS_EXIT_PERFORMANCE            ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     GRS_MODE                        ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     GRS_SYNCHRES                    ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     RACF_GRS_RNL                    ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     RACF_SENSITIVE_RESOURCES        ACTIVE(ENABLED)    EXCEPTION-HIGH     SYSB
     RSM_AFQ                         ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     RSM_HVSHARE                     ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     RSM_MAXCADS                     ACTIVE(ENABLED)    SUCCESSFUL         SYSB
     RSM_MEMLIMIT                    ACTIVE(ENABLED)    SUCCESSFUL         SYSB
```

SDSF: Sample CK Display

```
RSM_REAL                      20041006 07/15/2005 14:36:06  ONETIME ****
RSM_RSU                       20041006 07/15/2005 14:36:06  ONETIME ****
SDUMP_AUTO_ALLOCATION         20050118 07/15/2005 14:36:06  ONETIME ****
SDUMP_AVAILABLE               20050118 07/15/2005 14:36:06  ONETIME ****
USS_AUTOMOUNT_DELAY           20040808 07/28/2005 14:36:05    24:00 07/2
USS_FILESYS_CONFIG            20040217 07/28/2005 14:36:16    24:00 07/2
USS_MAXSOCKETS_MAXFILEPROC    20040808 07/28/2005 14:36:05    24:00 07/2
VSM_CSA_CHANGE                20040405 07/15/2005 14:36:06  ONETIME ****
VSM_CSA_LIMIT                 20040405 07/15/2005 14:36:06  ONETIME ****
VSM_CSA_THRESHOLD             20040405 07/29/2005 11:17:02     0:05 07/2
VSM_PVT_LIMIT                 20040405 07/15/2005 14:36:06  ONETIME ****
VSM_SQA_LIMIT                 20040405 07/15/2005 14:36:06  ONETIME ****
VSM_SQA_THRESHOLD             20040910 07/29/2005 11:21:14     0:15 07/2
```

SDSF: Sample CK Display, Browse a Check

```
IGVH107I The size of ECSA has not changed since the last IPL.

IGVH107I The size of EPVT has not changed since the last IPL.

IGVH500I
VSM_CSA_CHANGE
Virtual Storage Configuration Report
              Current IPL TOD:              Compare IPL TOD:
              07/15/2005 14:35:00.1468      06/30/2005 15:42:42.6894
              ------------------------      ------------------------
DATE          07/15/2005                    06/30/2005
TIME          14:35:00                      15:42:42
LOADxx        Z6                            Z6
IEANUC0x      1                             1
CSA()         Z6 (3200K,320M)               Z6 (3200K,320M)
SQA()         Z6 (6,256)                    Z6 (6,256)
FIX()         00 00                         00 00
LPA()         Z6 (Z6)                       Z6 (Z6)
MLPA()        Z6 Z6                         Z6 Z6
MLPA()        Z6 Z6                         Z6 Z6

Storage                    Current    Compare
Location    Change         Size       size          Start       End

PVT         0000000000    0000A00000 0000A00000    0000000000   0000A00000
                          (10M)      (10M)
CSA         0000000000    0000343000 0000343000    0000A00000   0000D43000
                          (3340K)    (3340K)
```

Simplifying operations – New z/OS management console* (Planned for 4Q 2005)

Value

- Simplify z/OS management for the new generation of IT professionals
- Automating, eliminating, and streamlining tasks
- Easily upgradeable to OMEGAMON® solutions

Planned Capabilities

- Task-oriented approach with GUI front end
- z/OS Health Checker data plus Tivoli Monitoring Services base capabilities
  - Expert Advice
  - Take Action
- Configuration status metrics for z/OS resources displayed using Tivoli Enterprise Portal
  - Improved ease-of-use of z/OS management
  - Value-add upgrades to comprehensive Tivoli Monitoring Services products

New product planned to be available in 4Q 2005 for no charge to z/OS customers

* All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

IBM Health Checker for z/OS Documentation

- User's Guide (SA22-7994-00)
- Will ship a sample check
  - Including sample message table