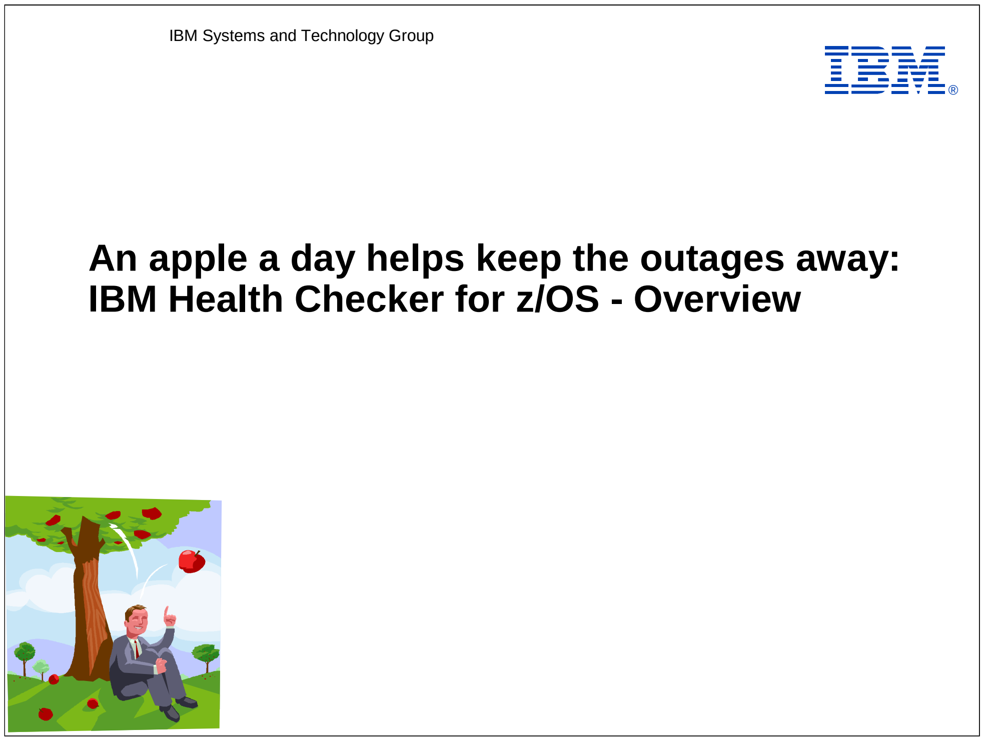
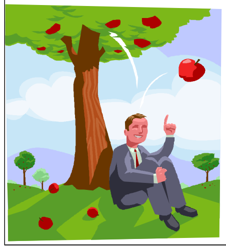


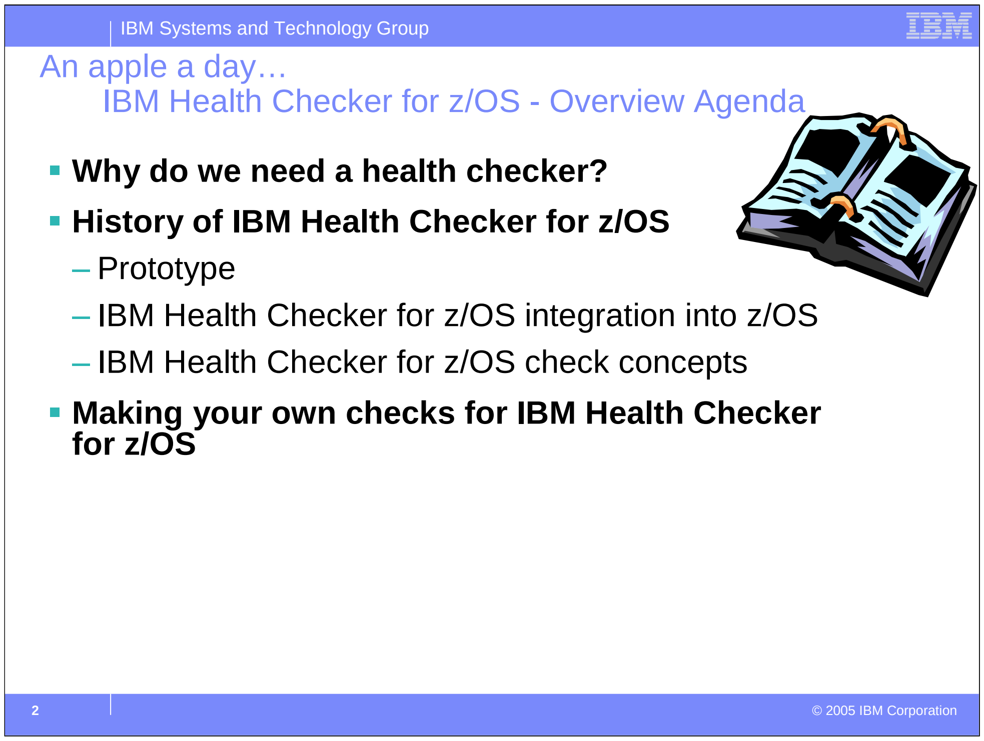
IBM Systems and Technology Group

An apple a day helps keep the outages away: IBM Health Checker for z/OS - Overview

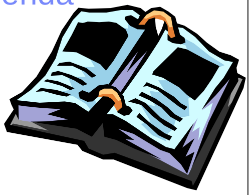


An apple a day helps keep the outages away: IBM Health Checker for z/OS - Overview

This session will discuss the very popular potential problem catcher - IBM Health Checker for z/OS. Originally delivered as a Web prototype, the IBM Health Checker for z/OS has matured into a z/OS R7 base function that is also available for z/OS R4, R5, and R6. It provides a very robust framework for automating the identification of potential problems.



An apple a day… IBM Health Checker for z/OS - Overview Agenda



- **Why do we need a health checker?**
- **History of IBM Health Checker for z/OS**
  - Prototype
  - IBM Health Checker for z/OS integration into z/OS
  - IBM Health Checker for z/OS check concepts
- **Making your own checks for IBM Health Checker for z/OS**

In this presentation we will discuss the following:

Why do we need a health checker?

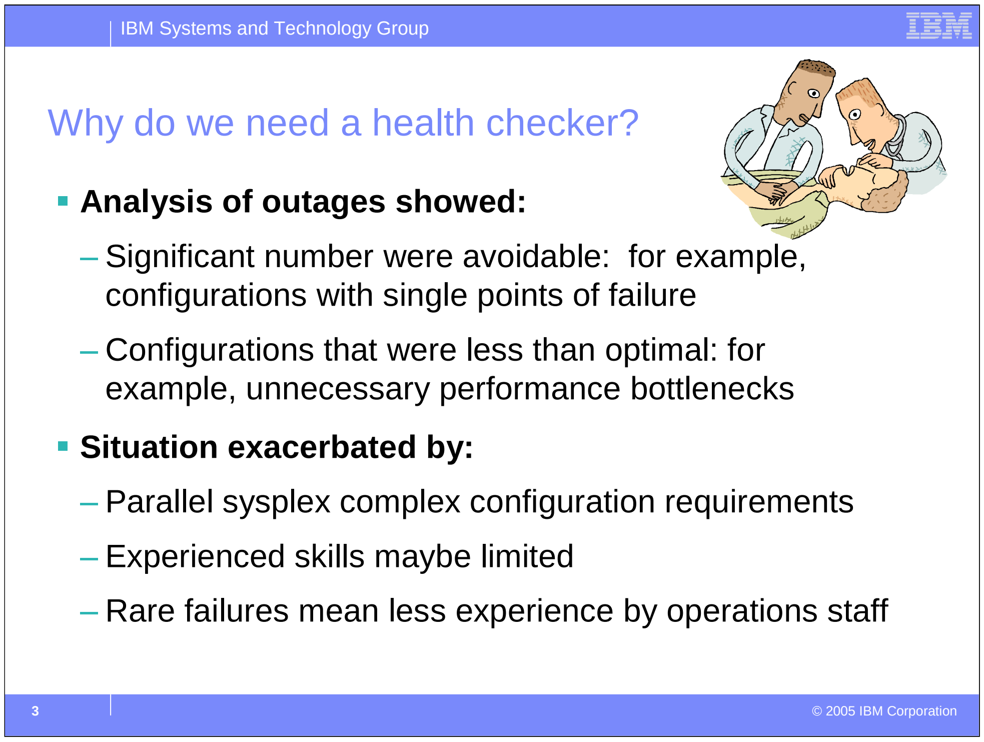
The history of IBM Health Checker for z/OS

The history of the health checker prototype

The integration of IBM Health Checker for z/OS into the system

We will also discuss IBM Health Checker for z/OS check concepts

We will show you how you can make your own checks for IBM Health Checker for z/OS



IBM Systems and Technology Group

Why do we need a health checker?

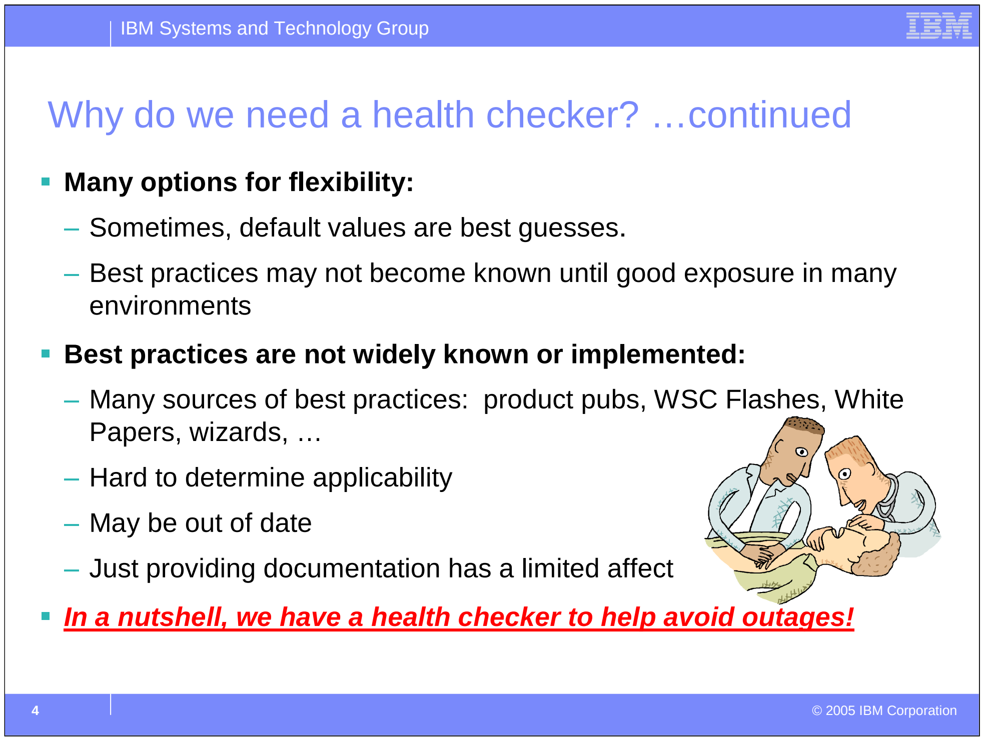
- **Analysis of outages showed:**
  - Significant number were avoidable: for example, configurations with single points of failure
  - Configurations that were less than optimal: for example, unnecessary performance bottlenecks
- **Situation exacerbated by:**
  - Parallel sysplex complex configuration requirements
  - Experienced skills maybe limited
  - Rare failures mean less experience by operations staff

3 © 2005 IBM Corporation

So why do we need a health checker?

Analysis of outages showed, that a significant number of outages were avoidable. For example configurations with single points of failure. Some configurations were less than optimal: for example, unnecessary performance bottlenecks.

The situation was exacerbated by: Configuration requirements for Parallel Sysplex complexes. In some cases limited access to experienced skills. And finally rare failures mean less experience by operations staff



Why do we need a health checker? …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 affect
- ***In a nutshell, we have a health checker to help avoid outages!***

Other reasons for having a heath checker are as follows. Often there are many options for flexibility:

Sometimes, default values are only best guesses.

Best practices may not become known until there has been good exposure in many different environments.

Also, once they are established, best practices are not widely known or implemented.

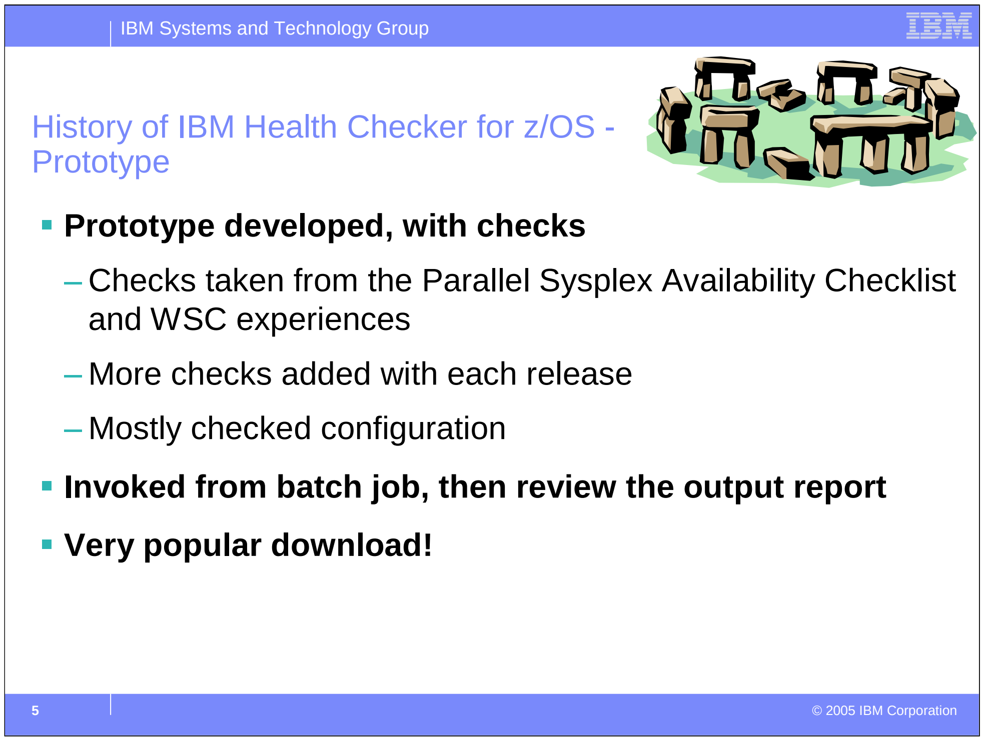
There are many sources of best practices: product pubs, WSC Flashes, White Papers, wizards, and so on.

Often is is hard to determine their applicability.

Occasionally, the best practice may be out of date.

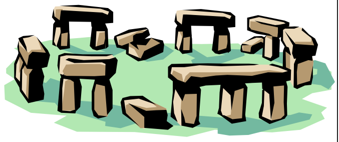
And finally, providing documentation alone has limited affect.

So in a nutshell, we have a health checker to help avoid outages!



IBM Systems and Technology Group

History of IBM Health Checker for z/OS - Prototype



- **Prototype developed, with checks**
  - Checks taken from the Parallel Sysplex Availability Checklist and WSC experiences
  - More checks added with each release
  - Mostly checked configuration
- **Invoked from batch job, then review the output report**
- **Very popular download!**

5

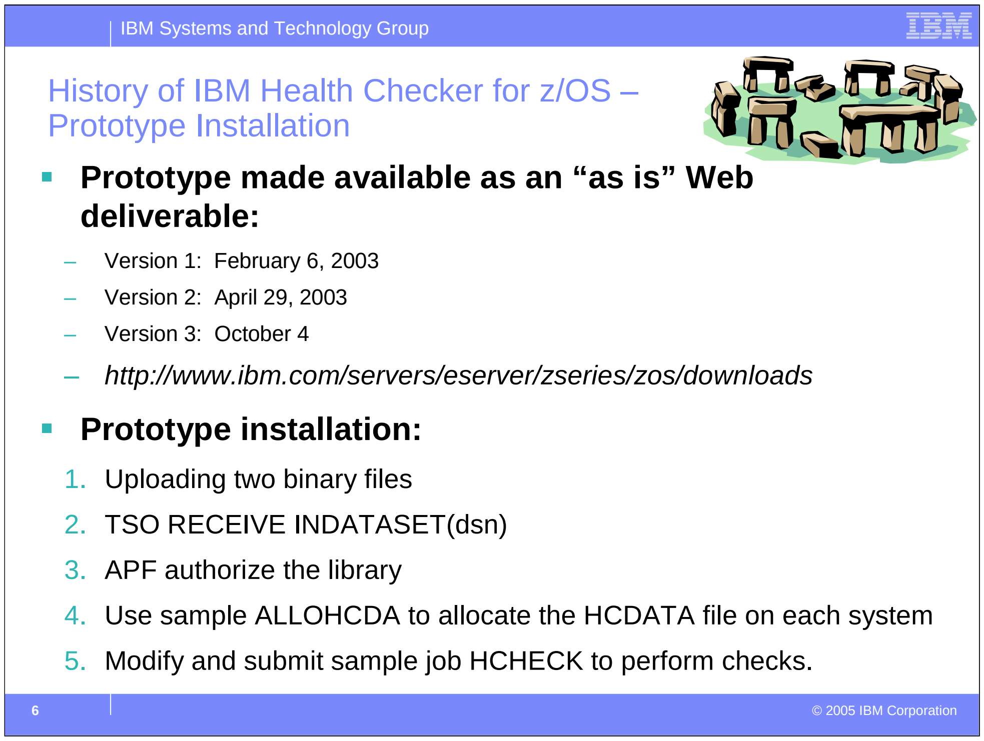
© 2005 IBM Corporation

Now for some history of Health Checker for z/OS prototype.

A prototype was developed, with checks. The checks were taken from the Parallel Sysplex Availability Checklist and Washington Systems Center experiences. More checks were added with each release. They mostly checked configuration

Heath Checker for z/OS is invoked via batch job, then review the output report

It was a very popular download!



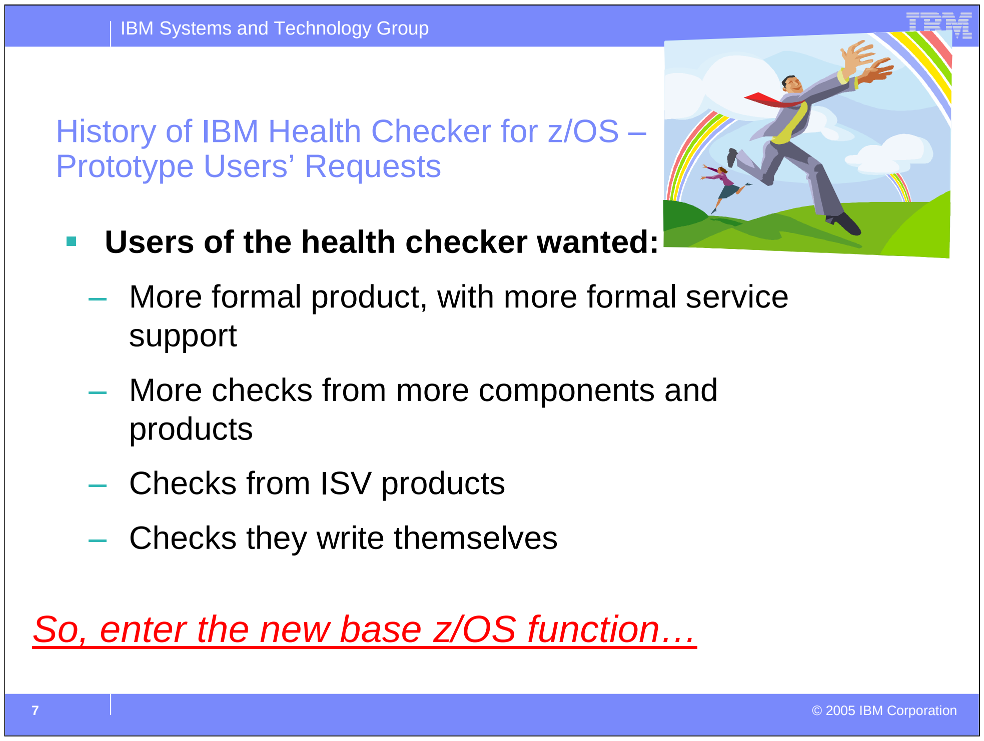
History of IBM Health Checker for z/OS – Prototype Installation

- **Prototype made available as an "as is" Web deliverable:**
  - Version 1: February 6, 2003
  - Version 2: April 29, 2003
  - Version 3: October 4
  - *http://www.ibm.com/servers/eserver/zseries/zos/downloads*
- **Prototype installation:**
  1. Uploading two binary files
  2. TSO RECEIVE INDATASET(dsn)
  3. APF authorize the library
  4. Use sample ALLOHCDA to allocate the HCDATA file on each system
  5. Modify and submit sample job HCHECK to perform checks.

The prototype was made available as an "as is" Web deliverable. Version 1, was first delivered on February 6, 2003. The second version was made available on April 29th, 2003, and version 3 was made available on October 4th at the following URL.

http://www.ibm.com/servers/eserver/zseries/zos/downloads

To install the prototype, you'll need to upload the two binary files use TSO RECEIVE INDATASET(dsn), then APF authorize the library Use the sample ALLOHCDA to allocate the HCDATA file on each system. Then modify and submit sample job HCHECK to perform checks.

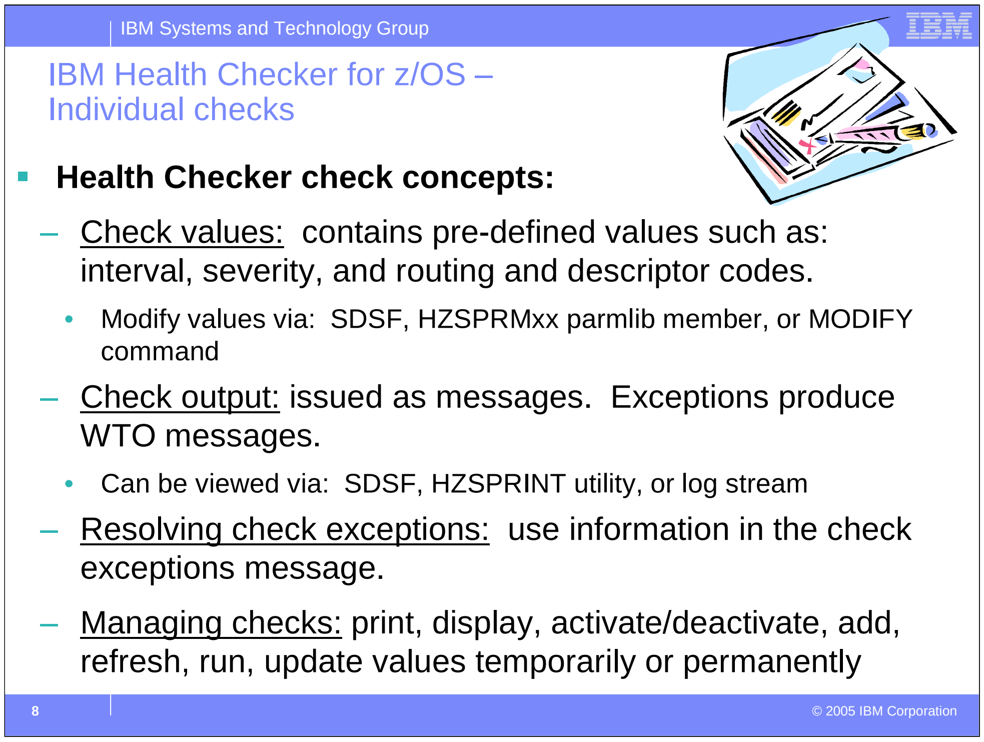


History of IBM Health Checker for z/OS – Prototype Users' Requests

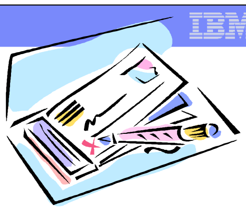
- **Users of the health checker wanted:**
  - More formal product, with more formal service support
  - More checks from more components and products
  - Checks from ISV products
  - Checks they write themselves

So, enter the new base z/OS function…

Users of the health checker prototype have requested a more formal product, with more formal service support. They requested more checks from more components and products, checks from ISV products. And they wanted to able to include checks they write themselves. So, enter the new base z/OS function.


IBM Systems and Technology Group

IBM Health Checker for z/OS – Individual checks



- **Health Checker check concepts:**
  - Check values: contains pre-defined values such as: interval, severity, and routing and descriptor codes.
    - Modify values via: SDSF, HZSPRMxx parmlib member, or MODIFY command
  - Check output: issued as messages. Exceptions produce WTO messages.
    - Can be viewed via: SDSF, HZSPRINT utility, or log stream
  - Resolving check exceptions: use information in the check exceptions message.
  - Managing checks: print, display, activate/deactivate, add, refresh, run, update values temporarily or permanently

8 © 2005 IBM Corporation

A check is actually a program or routine that identifies potential problems that may impact your availability or, in worst cases, cause outages. A checks is owned, delivered, and supported by the component, element, or product that writes it.

Checks are separate from the IBM Health Checker for z/OS framework. A check analyzes a configuration for in the following ways:

Changes in settings or configuration values that occur dynamically over the life of an IPL. Checks that look for changes in these values should run periodically to keep the installation aware of changes.

- Threshold levels approaching the upper limits, especially those that might occur gradually or insidiously.
- Single points of failure in a configuration.
- Unhealthy combinations of configurations or values that an installation might not think to check.

Check values: Each check includes a set of pre-defined values, such as:

- Interval, or how often the check will run
- Severity of the check, which influences how check output is issued
- Routing and descriptor codes for the check

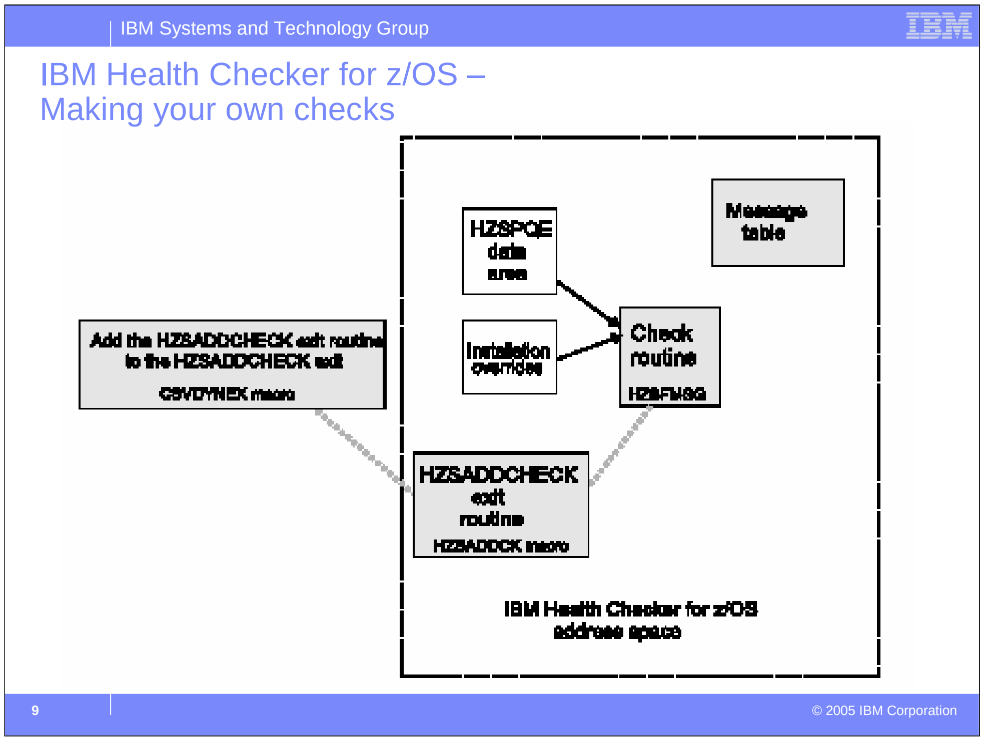
You can update or override some check values using either SDSF or statements in the HZSPRMxx parmlib member or the MODIFY command. These are called **installation updates**. You might do this if some check values are not suitable for your environment or configuration.

Check output: A check issues its output as messages, which you can view using SDSF, the HZSPRINT utility, or a log stream that collects a history of check output. If a check finds a potential problem, it issues a WTO message. We will call these WTO messages **exceptions**. The check exception messages are issued both as WTOs and also to the message buffer. The exception message in the message buffer includes both an explanation of the potential problem found, including the severity, as well as information on what to do to fix the potential problem.

Resolving check exceptions: To get the best results from IBM Health Checker for z/OS, you should let it run continuously on your system so that you will know when your system has changed. When you get an exception, you should resolve it using the information in the check exception message or overriding check values, so that you do not receive the same exceptions over and over.

Managing checks: You can use either SDSF, the HZSPRMxx parmlib member, or the IBM Health Checker for z/OS MODIFY (F *hzsproc*) command to manage checks. Managing checks includes:

- Printing check output from either SDSF, or using the HZSPRINT utility
- Displaying check information
- Take one time actions against checks, such as: activating or deactivating checks, adding new checks, refreshing checks (refresh processing first deletes a check from the IBM Health Checker for z/OS and then adds it back to the system), or running checks.
- Update check values temporarily using SDSF or the MODIFY *hzsproc* command.
- Update check values permanently using HZSPRMxx.



IBM Systems and Technology Group

IBM Health Checker for z/OS – Making your own checks

9

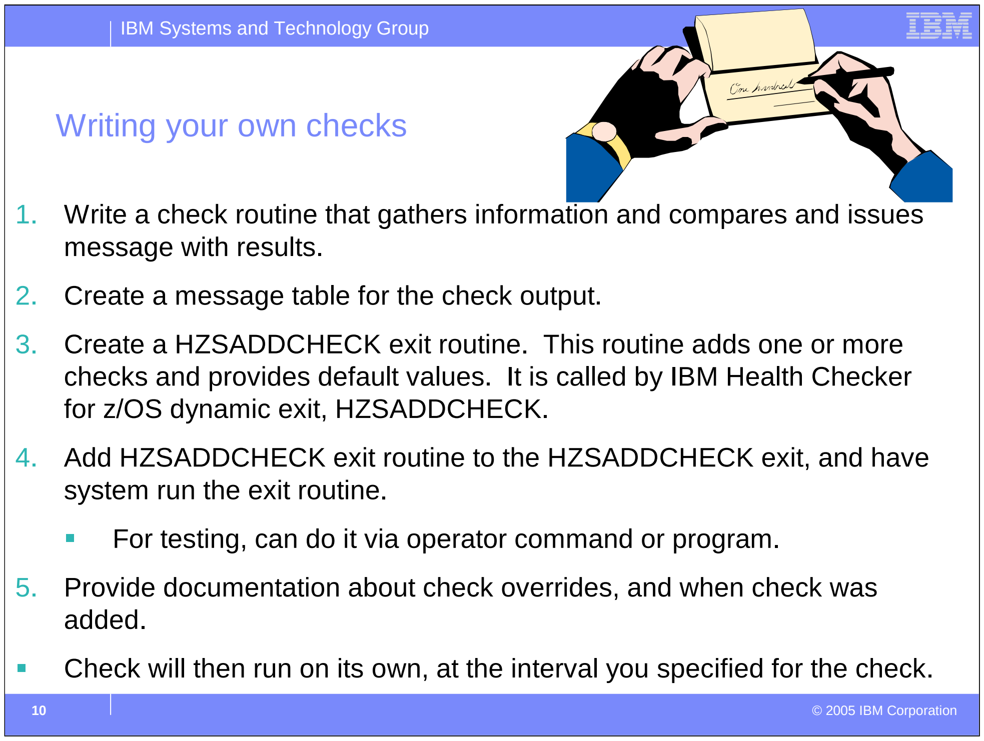
© 2005 IBM Corporation

For IBM Health Checker for z/OS checks, each check has three parts:

1) The dynamic exit routine that identifies the check to the IBM Health Checker for z/OS.
2) The check itself.
3) A message table to define messages that are issued by the check.

Each dynamic exit routine can either be added via an operator command or via an API (CSVDYNEX). Any dynamic exit routine that is added prior to the start of the IBM Health Checker for z/OS will be invoked when IBM Health Checker for z/OS is started. IBM Health Checker for z/OS must be told to run to pick up new checks that are added after IBM Health Checker for z/OS is started.

The foil above shows the parts of a check, and whether they run within the IBM Health Checker for z/OS address space or not. The shaded items show the parts that a check developer must provide.



Writing your own checks

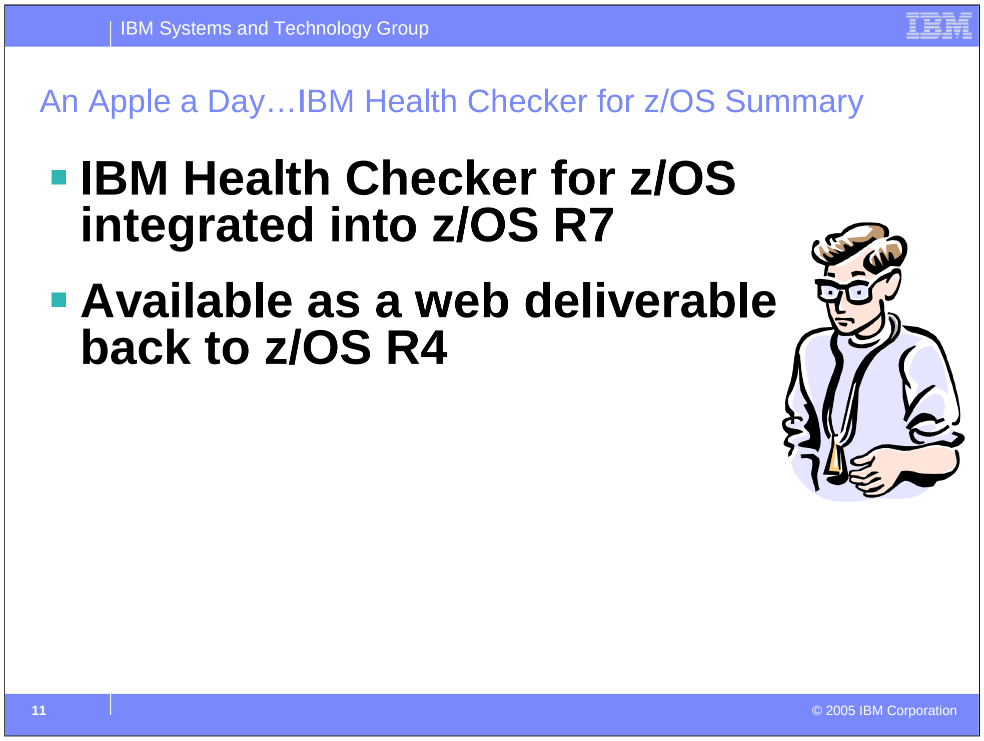
1. Write a check routine that gathers information and compares and issues message with results.
2. Create a message table for the check output.
3. Create a HZSADDCHECK exit routine. This routine adds one or more checks and provides default values. It is called by IBM Health Checker for z/OS dynamic exit, HZSADDCHECK.
4. Add HZSADDCHECK exit routine to the HZSADDCHECK exit, and have system run the exit routine.
   - For testing, can do it via operator command or program.
5. Provide documentation about check overrides, and when check was added.

- Check will then run on its own, at the interval you specified for the check.

To create a IBM Health Checker for z/OS check for your component or product, do the following:

1. Write a check routine that gathers information, compares current values with suggested settings or looks for configuration problems, and issues messages with the results of the check.
2. Create a message table for the check output. The message table defines the check output messages issued by the check routine.
3. Create a HZSADDCHECK exit routine. The HZSADDCHECK exit routine adds one or more checks, and, for each, provides the default values. This authorized routine runs in the IBM Health Checker for z/OS address space, called by the IBM Health Checker for z/OS dynamic exit, HZSADDCHECK.
4. Add the HZSADDCHECK exit routine to the HZSADDCHECK exit, and then have the system run the exit routines. The HZSADDCHECK exit routine adds checks to the IBM Health Checker for z/OS. For testing purposes, you can do this step with either operator commands or a program. However, when you package your check, you may want to add code to your product or component that can automatically look for and activate your checks at IPL time or when the product starts. When the exit routine adds your check(s) to the system, the system also applies any installation overrides to the default values for the check. .
5. Provide documentation about check-specific installation overrides to allow the installation to override the default check parameters defined when the check was added.

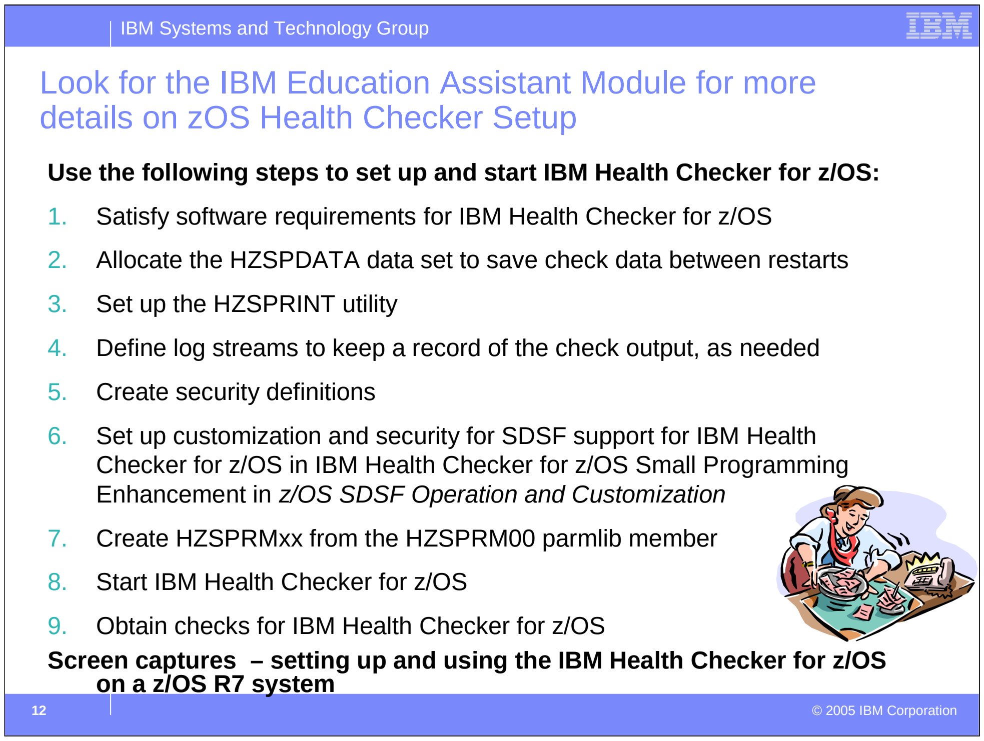
Once you've written a HZSADDCHECK exit routine, added it to the HZSADDCHECK exit, and the exit routine adds the check to IBM Health Checker for z/OS, the check will run on its own at the interval you specified for the check. IBM Health Checker for z/OS passes the check routine an HZSPQE data area containing check-specific information. The check routine will issue messages to report its results, whether it finds an exception to a suggested setting or not. The check continues to run at the interval defined for it, reporting any exception until the condition causing the exception is resolved.



An Apple a Day…IBM Health Checker for z/OS Summary

- **IBM Health Checker for z/OS integrated into z/OS R7**
- **Available as a web deliverable back to z/OS R4**

So, in summary IBM Health Checker for z/OS has been integrated into z/OS R7. And it has been made available as a web deliverable back to z/OS R4



Look for the IBM Education Assistant Module for more details on zOS Health Checker Setup

Use the following steps to set up and start IBM Health Checker for z/OS:

1. Satisfy software requirements for IBM Health Checker for z/OS
2. Allocate the HZSPDATA data set to save check data between restarts
3. Set up the HZSPRINT utility
4. Define log streams to keep a record of the check output, as needed
5. Create security definitions
6. Set up customization and security for SDSF support for IBM Health Checker for z/OS in IBM Health Checker for z/OS Small Programming Enhancement in *z/OS SDSF Operation and Customization*
7. Create HZSPRMxx from the HZSPRM00 parmlib member
8. Start IBM Health Checker for z/OS
9. Obtain checks for IBM Health Checker for z/OS

Screen captures – setting up and using the IBM Health Checker for z/OS on a z/OS R7 system

Look for the IBM Education Assistant Module for more details on z/OS Health Checker Setup

Use the following steps to set up and start IBM Health Checker for z/OS:

First, satisfy software requirements for IBM Health Checker for z/OS. Then

Allocate the HZSPDATA data set to save check data between restarts

Next set up the HZSPRINT utility

Then define log streams to keep a record of the check output, as needed

Create security definitions

Set up customization and security for SDSF support for IBM Health Checker for z/OS in IBM Health Checker for z/OS Small Programming Enhancement in *z/OS SDSF Operation and Customization*

Create HZSPRMxx from the HZSPRM00 parmlib member

Next, start IBM Health Checker for z/OS

And, obtain checks for IBM Health Checker for z/OS

Trademarks