* The objective of IBM Health Checker for z/OS is to identify potential problems before they impact your availability or, in worst cases, cause outages.
 - It checks the current active z/OS and sysplex settings and definitions for a system and compares the values to those suggested by IBM or defined by you.
 - It is not meant to be a diagnostic or monitoring tool, but rather a continuously running preventative that finds potential problems.
* IBM Health Checker for z/OS produces output in the form of detailed messages to let you know of both potential problems and suggested actions to take.
 NOTE:These messages do not mean that IBM Health Checker for z/OS has found problems that you need to report to IBM! IBM Health Checker for z/OS output messages simply inform you of potential problems so that you can take action on your installation.
* There are several parts to IBM Health Checker for z/OS:
 - The **framework** of the IBM Health Checker for z/OS is the interface that allows you to run and manage checks.
  > The framework is a common and open architecture, supporting check development by IBM, independent software vendors (ISVs), and users.
 - Individual **checks** look for component, element, or product specific z/OS settings and definitions, checking for potential problems.
  > The specific component or element owns, delivers, and supports the checks.
 - Checks can be either **local**, and run in the IBM Health Checker for z/OS address space, or **remote**, and run in the caller's address space.
 NOTE: Thus far, most IBM checks are local.
* A check is actually a program or routine that identifies potential problems before they impact your availability or, in worst cases, cause outages. A check 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 might analyze a configuration 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.
* A check is a program or routine that verifies that the current system environment is the most optimal and alerts an installation if any deviations are detected.
 - When a potential problem (an exception) is detected, it generates a check output, which is a report of messages that can help an installation to analyze the health of a system.

Check owner and check name: Each check has a check owner and check name.
* The check owner is the owning element or component. For IBM checks, checks, these will all start with IBM i.e. IBMASM and IBMUSS are two IBM check owners.
* The check name is the name of the check itself, such as ASM_NUMBER_LOCAL_DATASETS.

Check values: The values used by checks come from a variety of sources including product documentation and web sites.
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.

The output messages written to the message buffer can be viewed using:
* The SDSF CK panel
* The HZSPRINT utility
* A log history stream

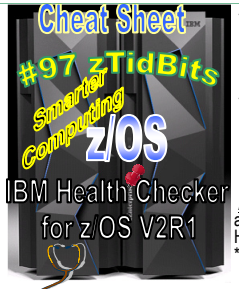
NOTE: You might find that the values that the check uses for comparison are not appropriate for your installation or for a particular system. If that is the case, you can either specify overrides to default values or suppress individual checks.
NOTE: You can update or override some check values using 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.

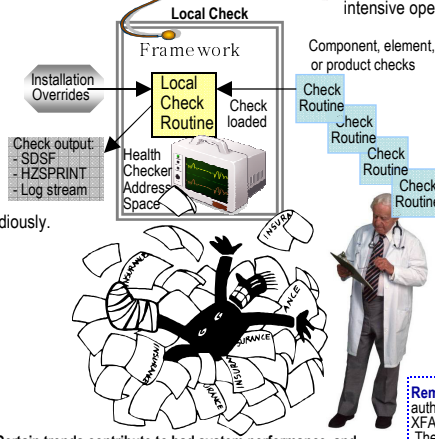
Check output: A check issues its output as messages, which you can view using SDSF, the HZSPRINT utility, or a log stream that collects a history of check output.
* If a check finds a potential problem, it issues a WTO message.
 - We will call these messages **exceptions**.
  Note: The check exception messages are issued both as WTOs and also to the message buffer. The WTO version contains only the message text, while the exception message in the message buffer includes both the text and 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
* Taking one time actions against checks, such as:
  – Activating or deactivating checks
  – Add new checks
  – Refresh checks - Refresh processing first deletes a check from the IBM Health Checker for z/OS and then adds it back to the system.
  – Run checks
* Updating check values temporarily using SDSF or the MODIFY hzsproc command.
* Updating check values permanently using HZSPRMxx.

Three types of checks are supported by IBM Health Checker for z/OS:
* A *local check* runs in IBM Health Checker for z/OS address space.
 Note: Most IBM checks provided so far are local checks.
* A *remote check* requires a dedicated task and runs in its address space.
* A *REXX check* runs in a System REXX address space in an APF authorized environment defined by System REXX.
 NOTE: REXX makes it easy to read data sets, parse retrieved information, and issue system commands.





Certain trends contribute to bad system performance and outage situations:
* IBM best practice recommendations are derived from different reference sources.
* While these sources are publicly available, defining appropriate recommendations requires time, experience, and specific skills.
* Recommendation changes due to new system requirements introduced in the system.
* Recommendation changes over time based on additional IBM or customer experiences.
* A *bad* system setting being in place for a long time, and it surfacing as a problem because new functions were enabled or certain events occurred on the system at the wrong time.
* Experienced z/OS system skills were limited at the installation.

A check can be written to analyze an installation configuration and look for these areas:
* Compliance to standards and consistent operating environments
* Changes in configuration values and settings that might have been set dynamically over the life of an IPL
* Threshold levels reaching the upper limit, especially those that might occur gradually.
* Single point of failure in a configuration
* Unhealthy combination of system settings or configuration values that an installer might not have thought to check
* Installation-related and migration-related actions
* Abnormal system behaviors
* A diagnostic tool running on the system that causes poor performances

Customer experiences
The following are examples of situations customers uncovered running IBM Health Checker for z/OS at different times:
* Configuration abnormalities in what was believed to be a stable system.
* Unexpected values on a system. Investigation revealed changes had been correctly made to that system, but not replicated on other systems.
* Default configurations that were never optimized for performance.
* Outdated settings that didn't support all current applications.
* Mismatched naming conventions that could have led to an outage.
* Dynamic changes accruing over the life of the IPL that can cause problems

A sample remote check can be found in SYS1.SAMPLIB.
• HZSSMSGT: sample message input
• HZSSRCHK: sample remote check routine
You can find additional check samples on the IBM Health Checker for z/OS website:
http://www.ibm.com/servers/eserver/zseries/zos/hchecker/

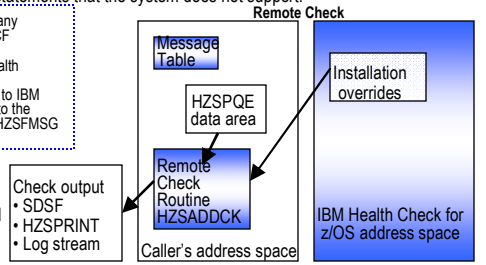
* The checks compare the system environment and parameters to established settings to uncover potential problems.
 - When z/OS IPLs, it automatically starts IBM Health Checker for z/OS, although it has a manual function if needed.
  Note: To stop and start IBM Health Checker for z/OS, use the HZSPROC started procedure in the following commands:
  STOP *hzsproc*
  START *hzsproc*,HZSPRM=PREV
* Specifying HZSPRM=PREV, which is the default in the HZSPROC procedure, ensures that you use the same set of HZSPRM*xx* parmlib members that were in effect in the previous instance of IBM Health Checker for z/OS.
 - You can prevent an automatic start of Health Checker at IPL time by assigning a special value of *NONE to the system parameter HZSPROC; for example the IEASYSxx parmlib member: HZSPROC=*NONE.
  NOTE: Installations might choose to use this feature to have better control on when certain address spaces, in this case Health Checker, start and use an automation product to issue an explicit START HZSPROC command at a desired time.

Anatomy of local checks: Local check is called with a parameter that points to the HZSPQE control block and a 4 K dynamic work area that is a data area containing all the information that a check routine needs, including the defaults defined in the HZSADDCHECK exit routine and any installation overrides.
* The PQE_FUNCTION_CODE tells the check why it is being called and ensures that the local check can access data from the IBM Health Checker for z/OS address space and that it does not require any potentially disruptive actions, such as I/O intensive operations, serializations, or waits.
 - A local check can use the HZSPDATA data set for persistent data that you want to save between restarts of the system or IBM Health Checker for z/OS. HZSPREAD and HZSPWRIT macros are used to read and write persistent data.
 - A local check invokes the HZSFMSG macro to issue messages and check results as described in the message table.

Anatomy of remote checks: have the following characteristics:
* Remote checks can run in any address space with any authority, and must have access granted by the RACF XFACILIT class profile.
 - The remote check must be registered with the IBM Health Checker where z/OS is running.
* A remote check can use the HZSPDATA data set for persistent data that you want to save between restarts of the system or IBM Health Checker for z/OS.
* A remote check invokes the HZSFMSG macro to issue a message with check results.

Prior to z/OS V2R1, there were no IBM Health Checker for z/OS related system parameters in IEASYSxx.
With z/OS V2R1, there are two system parameters, HZS and HZSPROC.
NOTE: If you share IEASYSxx parmlib members in a sysplex with systems that are at different levels ,you can make this work using the WARNUND system parameter, which tells the system to issue message IEA660I when it encounters unsupported IEASYSxx statements are encountered, rather than stopping an IPL to prompt for a correct statement. That means that a system can use an IEASYSxx member that includes statements that are not supported on the system's release level, simply ignoring the ones the system does not support. The system issues message IEA660I for IEASYSxx statements that the system does not support.

Remote checks can run in any address space with any authority, and must have access granted by the RACF XFACILIT class profile.
The remote check must be registered when IBM Health Checker for z/OS is running.
– A check issues the HZSADDCK macro to define it to IBM Health Checker for z/OS. The *handle* is assigned to the check to identify it for all remote HZSCHECK and HZSFMSG requests.



The illustration on the right displays:
* The IBM Health Checker for z/OS address space where the framework is running and communicating with the remote check in a separate caller's address space.
* The HZSPQE data area, which contains all the information that a check routine needs, including default values defined for the check and any installation overrides to those defaults.
* Installation overrides, which are changes that the installation can provide to check default values, such as intervals, parameters, and other values.
* The message table & the remote check routine provided by the check developer.

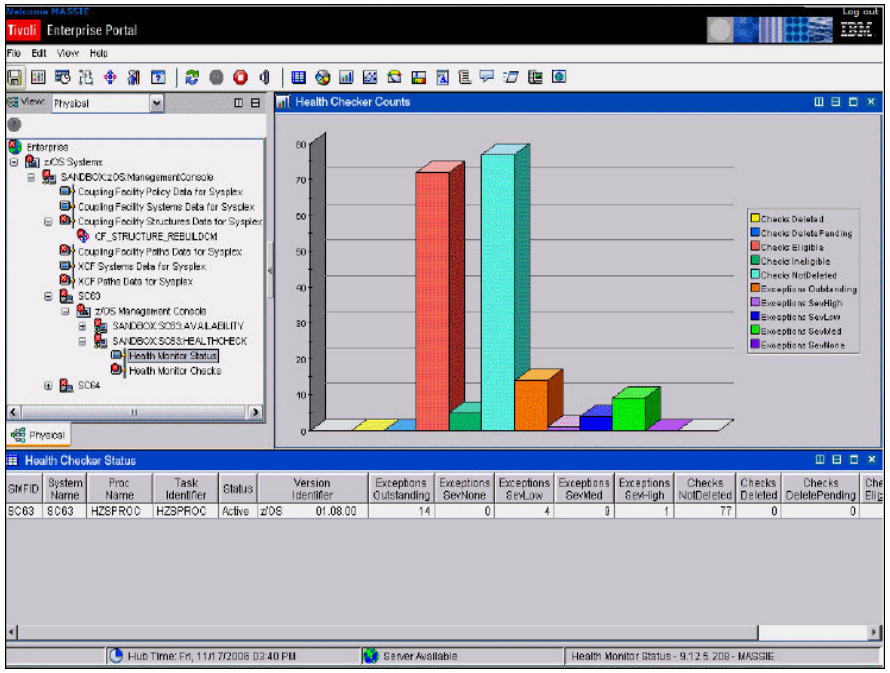
NOTE: A remote check can use the HZSPDATA data set for persistent data that you want to save between restarts of the system or IBM Health Checker for z/OS.

Types of check messages
When IBM Health Checker for z/OS executes on the system, the checks are run and produce output.
NOTE: The check message output indicates whether the check is successful.
The messages are categorized as follows:
Information messages
When a check executes and gets a clean run (no exception found) or a check is not appropriate in the current environment, the message text is written to the message buffer to indicate that the check was successful or that the check was not run.
Exception messages
When a check executes and detects a potential problem, it issues a write to operator (WTO) message, and the message is called an exception. The check exception messages are issued both as WTOs and to the message buffer. The WTO contains only the message text, whereas the message buffer includes the message text, an explanation of the potential problem uncovered, the severity, and a suggestion about what to do to fix the potential problem.
Reports
When a check finds an exception, it also writes a report to the message buffer with additional information for the exception message.
Debug
When a check executes in debug mode (DEBUG=ON), the debug messages are issued.
They are useful to diagnose a problem with the check.

The OMEGAMON z/OS Management Console provides two workspaces for accessing IBM Health Checker for z/OS:

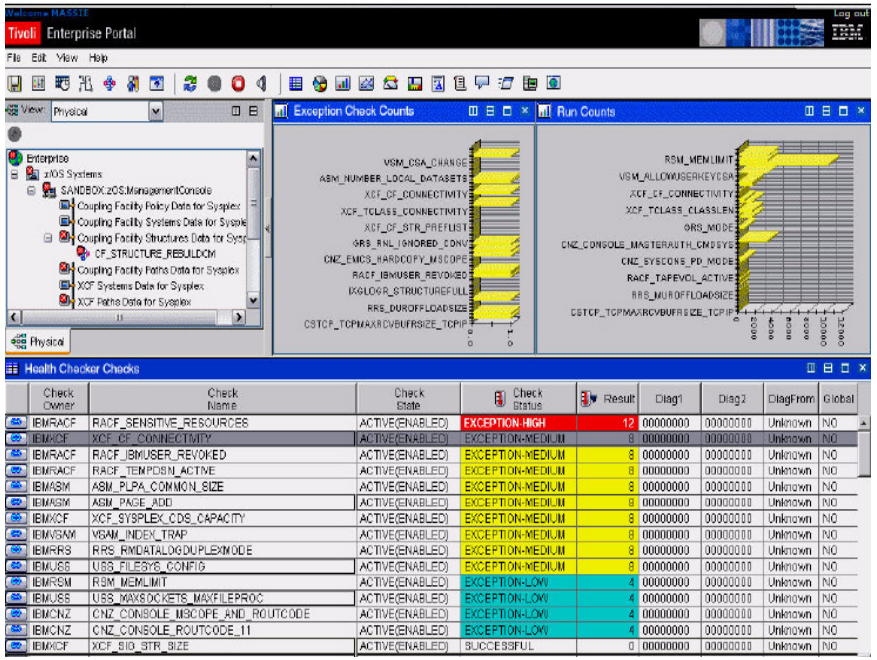
- Health Monitor Status
- Health Monitor Checks

The Health Monitor *Status* workspace provides general information about the IBM Health Checker framework and a statistical summary of the health checks. The figure below is an example of the Health Monitor Status workspace.



The IBM Health Checker Status frame shows the status of the IBM Health Checker address space and summary statistics about the health checks. The IBM Health Checker Counts frame shows a graph of the health check summary statistics.

The Health Monitor *Checks* workspace provides detailed information about the health checks.



The Exception Check Counts frame shows a graph of the health checks in EXCEPTION status statistics. The Run Counts frame shows a graph of the number of times that each health check has run statistics.

In the IBM Health Checker Checks frame, each health check is listed along with its status, such as EXCEPTION or SUCCESSFUL.
Each health check can be selected to display more detailed message information.