# z/OS V1R13

## IBM Health Checker for z/OS: HZS exception control

### Session objectives

- Describe newly available services and techniques related to health check exceptions
  - **DOM Control**
    - Check exception messages are typically WTOs, which need to be DOMed

    **Dynamic Severity**
    - Check exception messages have an assigned SEVERITY

    **Exception Interval**
    - Which can be different than the (wait-) INTERVAL after a “successful” health check run

### Overview – DOM control

- **Problem Statement / Need Addressed:**
  - Health check exception messages are typically sent to both the check's "message buffer" and to the console, the latter in form of a WTO (Write to Operator) message.
  - Previously, the system would always remove those WTOs in between check runs ("iterations"), just before a new iteration.
  - In certain situations this can lead to undesired, repeated alerts of the "same" potential problem.
- **Solution:**
  - While this automatic handling is good for most checks, more advanced health checks can now choose to decide on their own, when a DOM (Delete Operator Message) request is issued
  - If not DOMed, a previous WTO stays active, and the new (but typically duplicate) exception message just gets sent to the message buffer (with refreshed details)
- **Benefit:**
  - Duplicate (operator...) alerts are avoided

### Usage and invocation – DOM control

- A health check which wants to control the DOM requests for its own check exception messages, has to tell the system up front:
  - Existing service **HZSADDCK** (“Add check”) has a new optional DOM={SYSTEM|CHECK} parameter
  - The default, **DOM=SYSTEM**, preserves the existing behavior of the system always issuing DOM for any previous WTOs just before a new check iteration
  - **DOM=CHECK** tells the system to wait for the check to request the DOM and in the meantime suppress any WTOs for exception messages, if there are any outstanding, not-DOMed WTOs from a previous check iteration.
- During an iteration, a DOM=CHECK health check can now request an explicit DOM via new request type DOM of existing service HZSFMSG (“Format/Send a check message”):
  - **HZSFMSG REQUEST=DOM**
- A typical code flow in the check routine would be:
  - first determine, if this iteration's result is an exception or success
  - If success, clean up previous exception WTOs via HZSFMSG REQUEST=DOM and send the success message (those go only to the message buffer anyway)
  - If exception, and the previous iteration found an exception, too, determine, if this is basically “the same” exception
    - If so, do not request DOM and just use HZSFMSG as normal to send the exception message. The system will suppress a WTO and will only update the check's message buffer
    - If not “the same”, issue HZSFMSG REQUEST=DOM and then the “regular” HZSFMSG REQUEST=CHECKMSG...

### Overview – Dynamic severity

- **Problem Statement / Need Addressed:**
  - Previously, health checks could only have a single "severity" assigned to it (HIGH, MEDIUM, or LOW).
  - This severity determines the urgency for a resolution of any "exception" situation the check might detect.
  - For health checks where thresholds are involved, a fixed, single severity will **not let you gradually increase the severity** the closer a value gets to the threshold.
- **Solution:**
  - Dynamic Severity Control let's the check pick, at run time, an appropriate severity to use for each individual exception message
- **Benefit:**
  - More appropriate “urgency” levels for exception messages

### Usage and invocation – Dynamic severity

- A health check which wants to use “dynamic severity” for its check exception messages, has to tell the system up front:
  - Existing service **HZSADDCK** (“Add check”) has new optional AllowDynSev={NO|YES} parameter
  - The default, **AllowDynSev=NO**, preserves the existing behavior of the system using the fixed severity assigned to the check for any check exception messages. This SEVERITY is typically specified via HZSADDCK as well.
- **AllowDynSev=YES** allows the check to use the new **SEVERITY** parameter on the existing **HZSFMSG** (“Format/Send a message”) service:
  - HZSFMSG REQUEST=CHECKMSG … SEVERITY={SYSTEM|HIGH|MED|LOW|NONE|VALUE}
  - The default, SEVERITY=SYSTEM, will let the system use the SEVERITY as specified at Add Check time

  - The other values let the check specify which severity should be associated with this specific exception message
  - SEVERITY=VALUE, SEVERITYVAL=*severity* allows program variables, instead of hard-coded HIGH, MED, … severity values to be passed, avoiding long if-then-else's...
- An **AllowDynSev=YES** health check is expected to accept a special  syntax in its parameter string (parameter **PARMS** at “Add Check” time). This is to allow the installation to have influence on what dynamic severity to use, depending on system conditions.
- For example, assume a check which inspects system value XYZ which has a known maximum (threshold). A typical check parameter should then look like:
  - **PARMS==C'XYZ_HIGH(90%) XYZ_MED(75%) XYZ_LOW(50%)'**
- So the check will send
  - a HIGH severity exception message, when the inspected system value is 90% or higher of the allowed maximum
  - A MEDIUM severity exception message for the value being at 75% or higher, but less than 90%
  - … no exception, for the value being at less than 50%

## Overview – Exception interval

- **Problem Statement / Need Addressed:**
  - A health check, when not to be run only ONETIME, has an **INTERVAL** attribute, which specifies how long to wait to run again.
  - If the installation wishes a different schedule for when the check detects an exception, a separate **EXCEPTION INTERVAL** can be specified
  - Previously the EXCEPTION INTERVAL could only be **equal to or shorter than the (success-) INTERVAL**, even for situations where, for example, the systems programmer might need more time to “fix” the exception.
- **Solution:**
  - The system now allows an EXCEPTION INTERVAL value larger than the INTERVAL value. Previously the system “silently” capped the exception interval.
- **Benefit:**
  - Health checks can now have an INTERVAL short enough to be able to detect an exception in a timely manner, but can allow the installation more time to fix an exception, without having the check repeatedly alerting (paging) the operator.

## Interactions and dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - Check writers (DOM Control, Dynamic Severity)
  - Operators and System Programmers (Dynamic Severity, Exception Interval)

## Migration and coexistence considerations

- This function and associated syntax is only available in z/OS V1R13 and higher.
- Existing syntax will work as is and appropriate defaults will be chosen for any new values, guaranteeing unchanged behavior for existing checks.

## Installation

- The new function is part of the IBM Health Checker for z/OS, which is shipped with the base operating system.

## Session summary

- Advanced health checks can suppress additional exception message WTOs by delaying DOM requests for previous WTOs, thus **avoiding repeated, redundant alerts**
- Health Checks can use dynamic severities for their exception messages, as determined at runtime based on current system conditions, **allowing for more precise alerts**
- Longer EXCEPTION Intervals can help to **avoid repeated, redundant alerts**

## Appendix - References

- Related Publications
  - “IBM Health Checker for z/OS User's Guide” (SA22-7994)
    - Includes all the details for the new function and its associated syntax