Watchdogs and Daemons (Windows)

Piotr Bartkiewicz

MCS-1, DESY,Hamburg

Agenda

- PART 1: the MCS-1 Windows Watchdog
  - supervising crucial server applications
- PART 2: Console Daemon
  - automatic console screen management

PART I: the MCS-1 Windows Watchdog

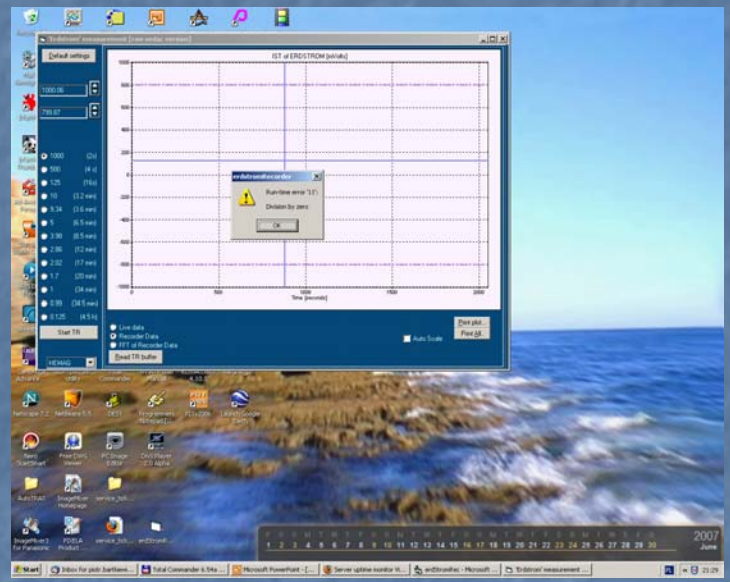
- Basic requirements for a software watchdog
- Some comments on application status and common misunderstandings
- The Heart Beat ActiveX component
- Functionality of the current watchdog version
- Remote control of the watchdog

Software Watchdog: the basic functionality

- Supervising the execution of other applications:
  - stopping the application, if an error occurred, or if the application is not in a *"proper"* state
  - automatic restart of the application, if the application is not running
  - notification about the restart and providing some information for further problem debugging

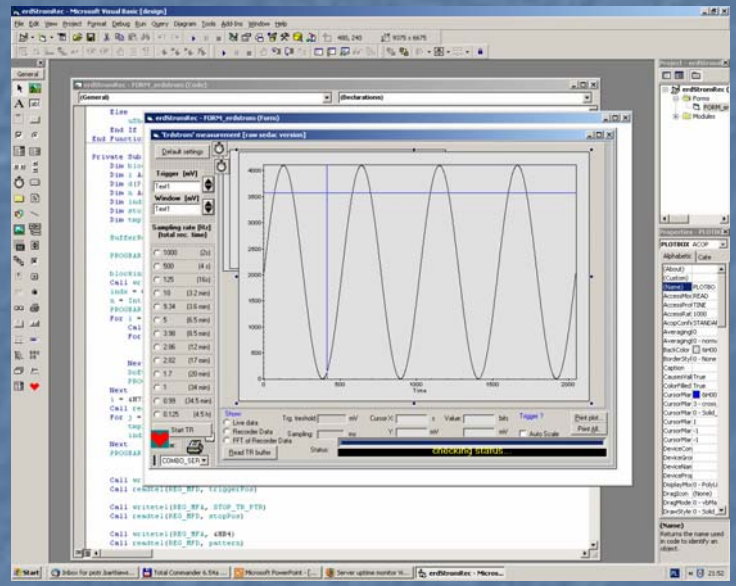
Comment on a “proper” application state

- Problem with applications having dialog-based error handlers:
  - application is waiting for the user input, showing an error box (mostly VB applications):
    - From the point of view of the operating system: this is not an error, application is still running
    - From the point of view of the application tasks: application is not running
- Possible solutions:
  - change the error handlers to make an application exit
  - use our Heart Beat ActiveX component (written especially for the MCS-1 Windows Watchdog)



The Heart Beat ActiveX component

- Designed to work with MCS-1 Windows Watchdog (uses the Windows Mailslot Interprocess Communication Mechanism)
- Remains idle, if no MCS-1 Windows Watchdog is present
- Easy in use: no program code modification needed, just in the IDE drop it on the application window
- Invisible during the run-time

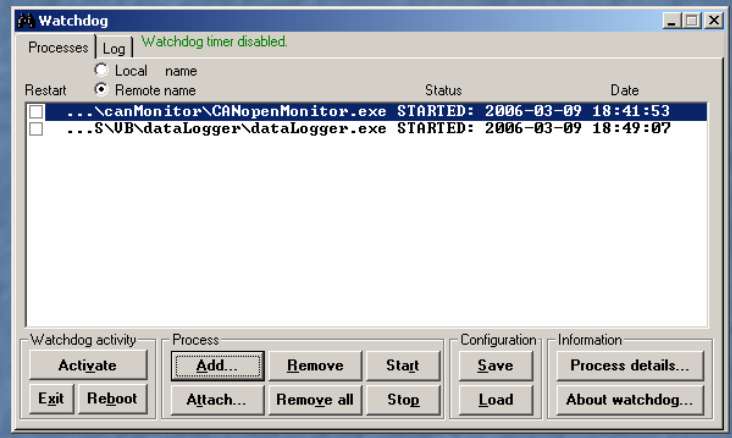


How the MCS-1 Windows Watchdog defines the “proper” application status

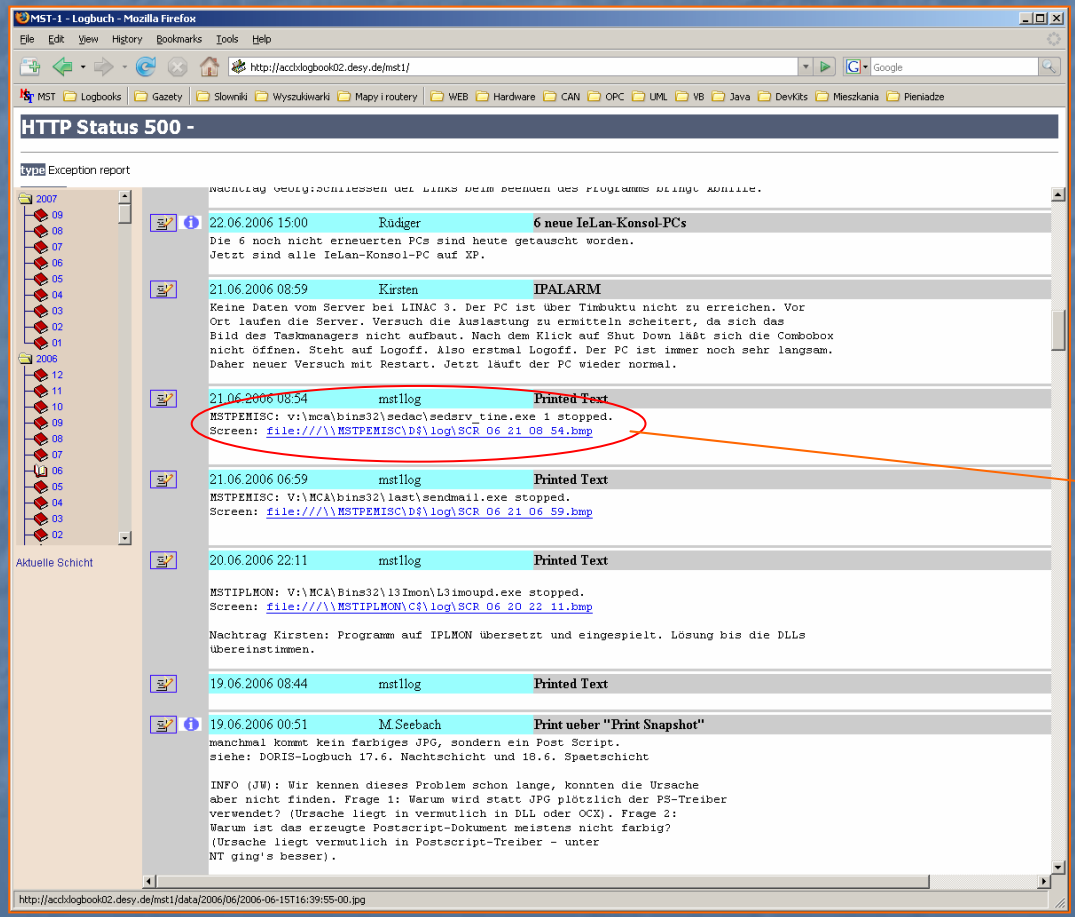
- application is running:
  - Windows OS shows ‘not signaled’ state of the application
- application is not “hanging”:
  - application responds for Windows Messages (sending WM_NULL to the application’s main window message queue)
- optional check: the heart beat is being updated (requires application having the Heart Beat component: hbeat.ocx)

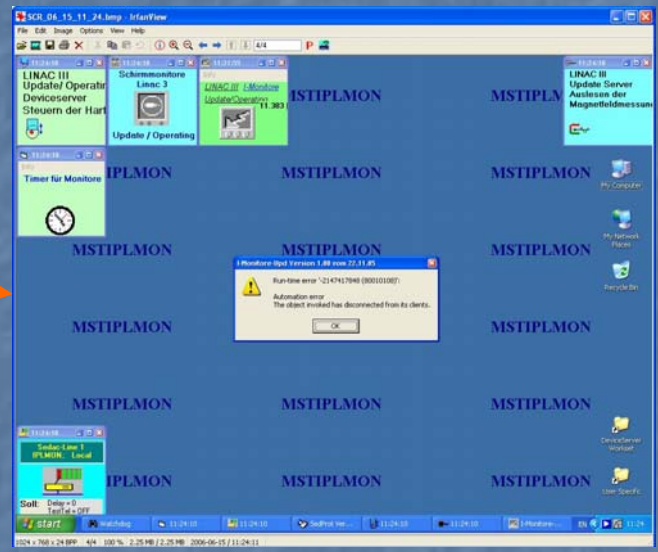
The current MCS-1 Windows Watchdog functionality

- Maintaining list of applications, which should be watched.
  - possibility of adding/remove applications, when watchdog is running
  - ASCII watchdog.cfg in $FEC_HOME configuration file stores list of supervised applications
- Stopping the application, which is not running “properly”
- Optionally automatic restart of application, which is considered as not running
- Logging information about each application start or stop.
  - Log files of ‘depth’ by default up to 3 days
  - Automatic entries in the MCS-1 logbook when stopping/restarting the application
  - Snapshot of the screen just before killing the application
- Possibility of remotely performing some operations like inspecting application status, application start and stop, machine reboot:
  - Watchdog registers itself as a TINE server with export name which is up to 13 last characters of a computer name + “_WD” (ex. MSTXPKAROL1_WD)

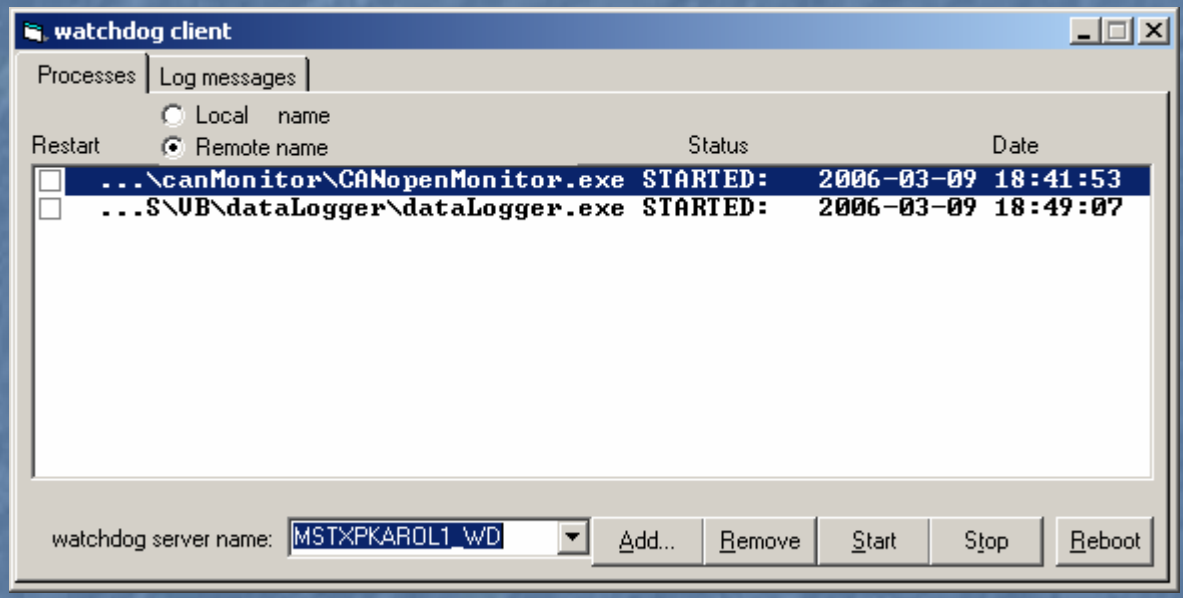


Automatic Watchdog Entries To The MCS-1 Logbook





Remote Control Of The Watchdog



PART II: The Console Daemon: automatic console screen management

- Motivation on the example of HERA consoles
- Idea of the automatic console screen management
- Implementation of the Console Daemon
- Remote screen management tools

Accelerators control room: HERA consoles

- 20 operational steps needed to reach the luminosity run
- 25 Windows XP consoles
- more then 150 application
- all applications can be started from any workstation
- no firm standard for window application size
- machine subsystem-specific applications are written by various group of engineers

Important for the shift crew efficiency:

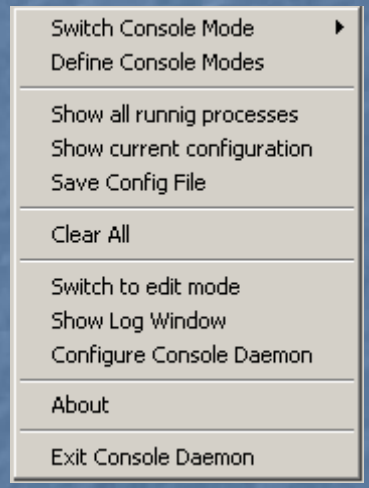
- applications are started just before they are needed
- not longer needed applications are terminated
- applications appear on the same consoles with the same window position and geometry

The Idea How To Automatically Manage Console Screens

- on each console workstation run a dedicated TINE server, a ‘Console Daemon’, which makes a console a member of a centrally managed system
- make the Console Daemon able to start and stop any of available application, and manipulate the windows geometry
- create groups of applications and assign them to accelerator states, store groups in Console Daemons’ configuration files
- drive each Console Daemon by the TINE client applications:
  - HERA Sequencer
  - Command line interface (scripts, desktop shortcuts)
  - Remote Console Screen Manager tool

Implementation Of The Console Daemon [1]

- A TINE windowless server (to be seen in a System Tray)
- Two working modes:
  - Operational
  - Editing

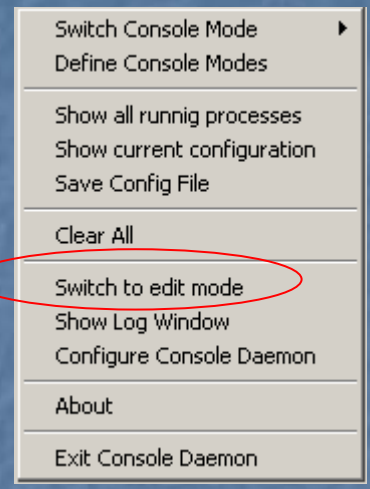


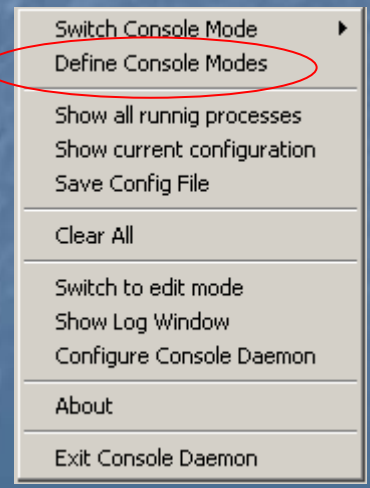
Implementation Of The Console Daemon [2]

- Operational Mode:
  - just after entering operational mode reads locally stored XML configuration file containing:
    - definitions of application groups (console modes)
    - for each console mode: groups of applications (name, path, command line arguments, window captions, and geometry)
  - When selecting a console mode:
    - stopping all applications which do not belong to that mode
    - starting applications belonging to the mode
    - placing windows in desired locations and resizing to specified dimensions.
    - optionally:
      - for very important applications, make an automatic restart , if the application has crashed
      - keep windows geometry and locations during the lifetime of the application.

Implementation Of The Console Daemon [3]

- Editing mode:
  - used to:
    - define console modes (like injection, ramp, luminosity run etc.)
    - build the group of applications for each console mode,
    - specify arguments for applications
    - define windows geometry
  - Configuration process:
    - create a console mode and assign a name to it, or select an existing console mode
    - start all requested for the console mode applications, stop those not belonging to the current console mode
    - place and resize all windows
    - save the configuration file (locally stored XML file)



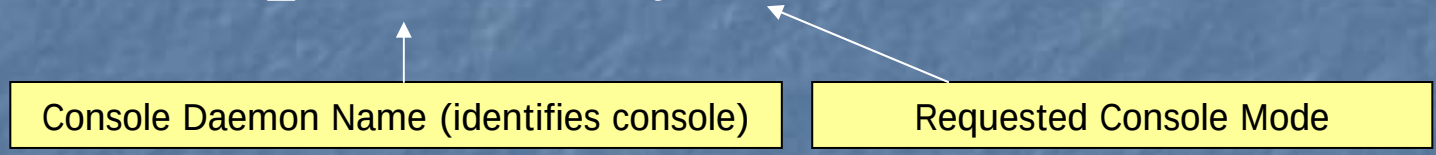


Remote Console Screen Management Tools

- Simple command line programs, which drives the console mode switches (to be used in batch files, scripts and desktop shortcuts).

  Example:

  setconsmode CD_HERACON09 e-injection



- Remote Console Screen Manager

Remote Console Screen Manager