RELEASE NOTES

CMON-Line: Monitoring V1.0
for VME/VPX Blades and TRACe Computers
Kehm Package ID 17202

SD.DT.G78-1e - January 2018

POSSIBILITIES START HERE kontron

CMON-Line Release Notes

Disclaimer

Kontron would like to point out that the information contained in this user guide may be subject to alteration, particularly as a result of the constant upgrading of Kontron products. This document does not entail any guarantee on the part of Kontron with respect to technical processes described in the user guide or any product characteristics set out in the user guide. Kontron assumes no responsibility or liability for the use of the described product(s), conveys no license or title under any patent, copyright or mask work rights to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement unless otherwise specified. Applications that are described in this user guide are for illustration purposes only. Kontron makes no representation or warranty that such application will be suitable for the specified use without further testing or modification. Kontron expressly informs the user that this user guide only contains a general description of processes and instructions which may not be applicable in every individual case. In cases of doubt, please contact Kontron.

Kontron S&T AG

Lise-Meitner-Str. 3-5

86156 Augsburg

Germany

www.kontron.com

Revision History

PUBLICATION TITLE:	CMON-Line Release Notes	
DOC. ID:	SD.DT.G78-1e	
Revision	Brief Description of Changes	Date of Issue
1e	Updated sections: - 1/ Introduction - 2/ Benefits - 3.3.4 CMON-Line Eval and Development Versions	01-2018
0e	Initial Issue	09-2017

Customer Support

Please contact our support team at support.KFR@kontron.com

Customer Service

As a trusted technology innovator and global solutions provider, Kontron extends its embedded market strengths into a services portfolio allowing companies to break the barriers of traditional product lifecycles. Proven product expertise coupled with collaborative and highly-experienced support enables Kontron to provide exceptional peace of mind to build and maintain successful products.

For more details on Kontron's service offerings such as: enhanced repair services, extended warranty, Kontron training academy, and more visit http://www.kontron.com/support-and-services/services.

Customer Comments

If you have any difficulties using this manual, discover an error, or just want to provide some feedback, contact Kontron support. Detail any errors you find. We will correct the errors or problems as soon as possible and post the revised manual on our website.

Symbols

The following symbols may be used in this manual:

⚠DANGER	DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.
⚠WARNING	WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.
⚠CAUTION	CAUTION indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	NOTICE indicates a property damage message.
	**Electric Shock!** This symbol and title warn of hazards due to electrical shocks (> 60 V) when touching products or parts of them. Failure to observe the precautions indicated and/or prescribed by the law may endanger your life/health and/or result in damage to your material.
	**ESD Sensitive Device!** This symbol and title inform that the electronic boards and their components are sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.
	**HOT Surface!** Do NOT touch! Allow to cool before servicing.
	**Laser!** This symbol inform of the risk of exposure to laser beam from an electrical device. Eye protection per manufacturer notice shall review before servicing.
	This symbol indicates general information about the product and the user manual. This symbol also indicates detail information about the specific product configuration.
	This symbol indicates important information which must be read carefully.
	This symbol precedes helpful hints and tips for daily use.

For Your Safety

Your new Kontron product was developed and tested carefully to provide all features necessary to ensure its compliance with electrical safety requirements. It was also designed for a long fault-free life. However, the life expectancy of your product can be drastically reduced by improper treatment during unpacking and installation. Therefore, in the interest of your own safety and of the correct operation of your new Kontron product, you are requested to conform with the following guidelines.

High Voltage Safety Instructions

As a precaution and in case of danger, the power connector must be easily accessible. The power connector is the product's main disconnect device.

⚠CAUTION

Warning!
All operations on this device must be carried out by sufficiently skilled personnel only.

⚠CAUTION

Caution, Electric Shock!
Before installing a non hot-swappable Kontron product into a system always ensure that your mains power is switched off. This also applies to the installation of piggybacks. Serious electrical shock hazards can exist during all installation, repair, and maintenance operations on this product. Therefore, always unplug the power cable and any other cables which provide external voltages before performing any work on this product.
Earth ground connection to vehicle's chassis or a central grounding point shall remain connected. The earth ground cable shall be the last cable to be disconnected or the first cable to be connected when performing installation or removal procedures on this product.

Special Handling and Unpacking Instructions

ESD Sensitive Device!
Electronic boards and their components are sensitive to static electricity. Therefore, care must be taken during all handling operations and inspections of this product, in order to ensure product integrity at all times

Do not handle this product out of its protective enclosure while it is not used for operational purposes unless it is otherwise protected.

Whenever possible, unpack or pack this product only at EOS/ESD safe work stations. Where a safe work station is not guaranteed, it is important for the user to be electrically discharged before touching the product with his/her hands or tools. This is most easily done by touching a metal part of your system housing.

It is particularly important to observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings etc. If the product contains batteries for RTC or memory backup, ensure that the product is not placed on conductive surfaces, including anti-static plastics or sponges. They can cause short circuits and damage the batteries or conductive circuits on the product.

General Instructions on Usage

In order to maintain Kontron's product warranty and CE compliance, this product must not be altered or modified in any way. Changes or modifications to the product, that are not explicitly approved by Kontron and described in this manual or received from Kontron's Technical Support as a special handling instruction, will void your warranty and the CE compliance.

This product should only be installed in or connected to systems that fulfill all necessary technical and specific environmental requirements. This also applies to the operational temperature range of the specific board version, that must not be exceeded. If batteries are present, their temperature restrictions must be taken into account.

In performing all necessary installation and application operations, only follow the instructions supplied by the present manual.

Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the product then re-pack it in the same manner as it was delivered.

Special care is necessary when handling or unpacking the product. See Special Handling and Unpacking Instruction.

Environmental Protection Statement

This product has been manufactured to satisfy environmental protection requirements where possible. Many of the components used (structural parts, printed circuit boards, connectors, batteries, etc.) are capable of being recycled.

Final disposition of this product after its service life must be accomplished in accordance with applicable country, state, or local laws or regulations.

Environmental protection is a high priority with Kontron.
Kontron follows the DEEE/WEEE directive.
You are encouraged to return our products for proper disposal.

The Waste Electrical and Electronic Equipment (WEEE) Directive aims to:

- Reduce waste arising from electrical and electronic equipment (EEE)
- Make producers of EEE responsible for the environmental impact of their products, especially when they become waste
- Encourage separate collection and subsequent treatment, reuse, recovery, recycling and sound environmental disposal of EEE

Improve the environmental performance of all those involved during the lifecycle of EEE

Terms and Conditions

Kontron warrants products in accordance with defined regional warranty periods. For more information about warranty compliance and conformity, and the warranty period in your region, visit http://www.kontron.com/terms-and-conditions.

Kontron sells products worldwide and declares regional General Terms & Conditions of Sale, and Purchase Order Terms & Conditions. Visit http://www.kontron.com/terms-and-conditions

For contact information, refer to the corporate offices contact information on the last page of this user guide or visit our website CONTACT US.

Table Of Contents

List Of Figures

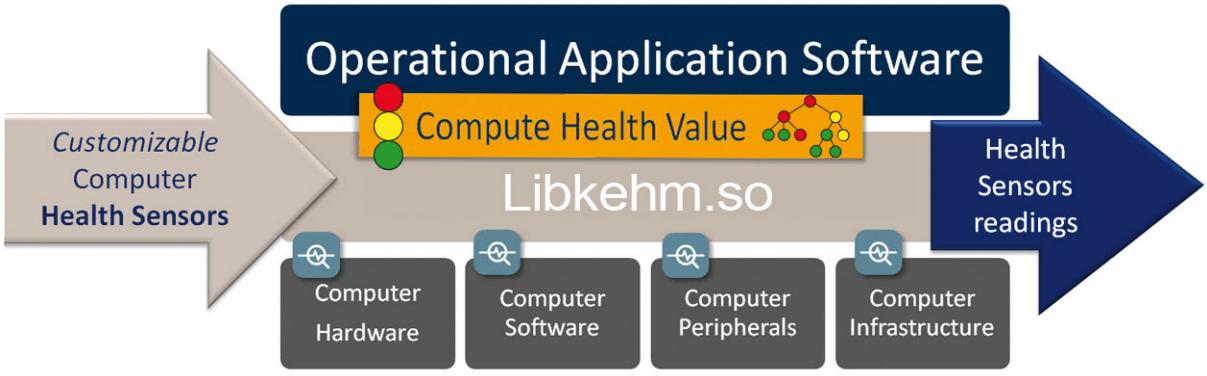
1 / Introduction

Monitoring the behavior of an embedded computer requires checking many measurement points simultaneously. Moreover, this challenge is made even more difficult with open architectures, where multiple customization stages of hardware and software can alter the computer after it was checked OK on the manufacturing line. To top that, each embedded application use case is different and requires specific attention to specific computer resources.

Figure 1: Computer Health Sensor possible Values

Kontron monitoring (kehm) is an open, data driven framework designed to extend basic computer health information to cover all customized elements (hardware, software, infrastructure) so as to issue a single pass health status in the form of a health sensor tree which top level SUCCESS, WARNING or ALARM value represents the complete solution status.

Figure 2: CMON-Line Monitoring Overview using Kehm library



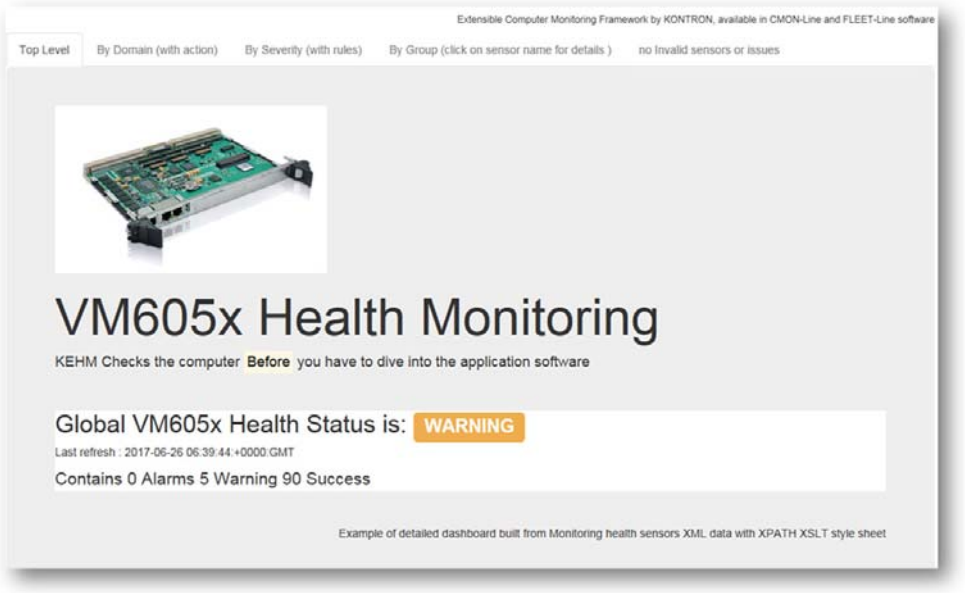
Monitoring comes as a library and sample files. The library computes health sensor values from a customizable health sensors definition file, and provides this health values in the form of an object tree, with the complete solution health at the top , and all health sensors information in the health tree branches and leaves.

2 / Benefits

After installing and configuring CMON-Line Monitoring for your platform (10 mn work, see chapter 3 - Getting Started) you can enjoy a first CBIT service, reporting alerts in the system journal as well as providing periodic detailed health information. This information is fed to a XML output file, which contains both health sensors definition and current sensor readings and health value. XML files are ready to be processed by both human beings and programs alike. CMON-Line monitoring uses the same XML structure for input and output files.

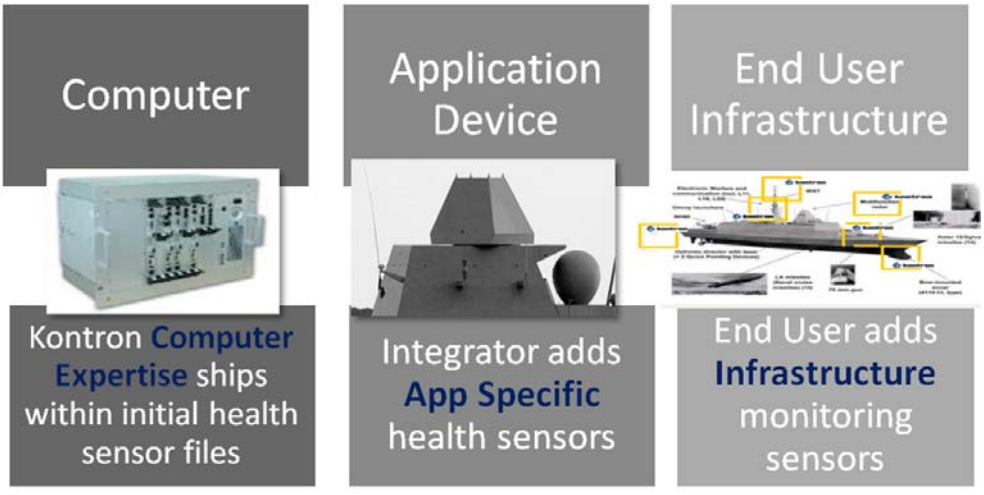
To enjoy the power of a data driven framework, examples of dashboards (see section 3.3.2 page 4) are also provided, which you can use from the same machine, or from another computer in the same network.

Figure 3: Top level view of computer health, built on the fly from kehm XML result file.



Customizing the computer health monitoring with extra sensor definitions for your own measurement points just need a text editor. The modified file becomes the new computer health assessment reference and covers your complete solution. Delivering this file and kehm monitoring to the final customer along with the system tremendously helps post deployment troubleshooting and support.

Figure 4: Via CMON-Line Monitoring Sensor Files, the system health expertise travels with the computer



With built in extensibility, kehm monitoring is the right solution to leverage the expertise of all actors of the solution supply chain, adding specific knowledge at each step, up to the final commissioning of the system by the end user into his infrastructure.

3 / Getting Started

3.1 First Step

The typical first experience with KEHM, looks like this (example taken on VM605x):

```
• tar xvf kehm-eval-1.0-17202.tar
• dnf --repofrompath KEHM,. install kehm-VM605x
• cd /usr/share/kehm
• ./kehm_example -o -c &
  (wait some seconds for kehm-RESULT.xml to be created)
• firefox kehm-RESULT.xml &
  (Use the menu to navigate to sensors evaluation details....)
```

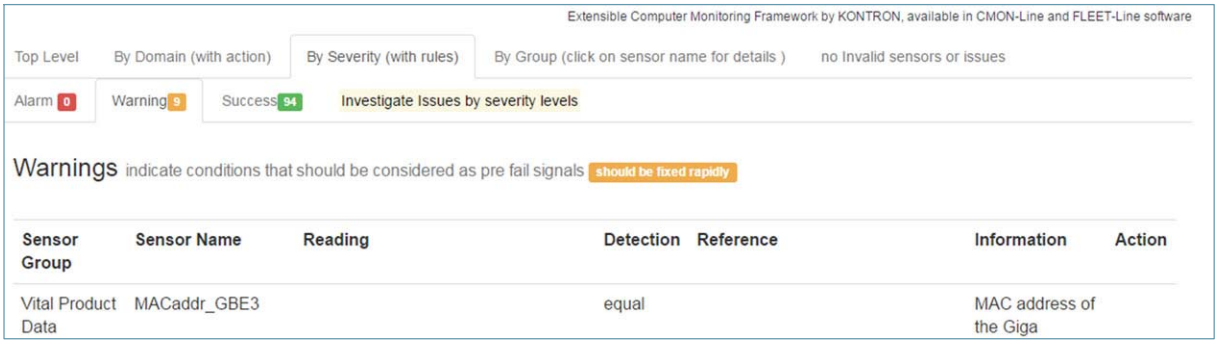
The main steps being:

- Installation: Using **dnf**, install the software for your platform from the repository or from the directory where the KEHM **rpms** have been stored.
- First try: Go to **/usr/share/kehm**, and attempt a first computer health assessment using **kehm_example.exe** with **[Platform]_sensors.xml** as the input file. (choose the closest platform to your hardware: TRACe, VM60xx or VX30xx). See kehm_example.exe(1) for more details - section 4.3 page 14.
- Simple processing of the result file: **kehm_walktree.exe** is another sample program , ready to display the computer health result at any depth of the health three on a terminal window. See **kehm_wallktree**(1) for information - section 4.4 page 16.
- Continuous monitoring: the **kehmcbit** service file is prepared to implement a continuous computer health assessment using **kehm_example** and **systemctl**. See **kehmcbit.service** - section 4.5 page 17.
- Remote monitoring dashboards, simple http service: to analyse the health information, the best solution is to transform the XML output into html code and visualize it using an http browser. This is done on the fly using XSL and XPATH technologies. Two examples of dashboards are done with **kehm-devel.xsl** and **kehm-pretty.xsl** - section 4.7 page 19.

If no http service is already running on your target platform, using **kehmhttp** service will leverage the SimpleHTTPServer python library module. This is explained in detail in **kehmhttp.service** - section 4.6 page 18.

Adapting to your Hardware

Figure 5: Kehm 'pretty' Dashboard: 'by Severity' View



Since the example sensor files include deep computer details (Serial number, Variant, Order Code, BIOS rev etc), some kehm sensors will have the warning status, which will propagate to the top level. You can easily find them using the

'pretty' dashboard and use the 'By Severity' view and see what is different between the reference (in the file) and the current sensor reading.

One other change likely needed is in the temperature section. Temperature references have been set for the standard air cooled version of the platform (SA version) . If you use extended air cooled (WA, RA versions) or conduction cooled (RC version) computers, you can operate the computer in a wider range and sensor warning and alarm conditions should be changed accordingly.

3.2 Going Further

3.2.1 Multiple Situations where Kehm can help

With an extensible data driven framework , kehm monitoring can be customized to help multiple actors of the embedded computing market. The **kehm** man page (section 4.1 page 6) describes some kehm use case like CBIT, Acceptance tests, Support and Complex supply chain. **Kehm-sensors** (section 4.2 page 9) also cover the system characterization use case (see the minValue, maxValue and the various xxCount attributes).

3.2.2 Understand the KEHM Health Sensor Data Model

Kehm-sensors (section 4.2 page 9) manual page includes all the information regarding each property of the kehm health sensor and their use. The data model includes many possibilities for customization to match most use cases. Understand the use of sensors attributes that are not used by the kehm library, use those fields the way you want to tag , filter and sort information to your taste, this will not impact the library nor the examples. You will just find them untouched in the output file, with the exact same content as in the input file.

3.2.3 Extend the Health Monitoring Coverage

Once you understand the health sensor data record, it is easy to create more sensors to cover your specific system. However, to run modified XML sensor files, the full version of KEHM is needed (contact support.KFR@kontron.com).

3.3 Even Further

3.3.1 Logical Sensors

Logical sensors are powerful health measurement tools which allow multiple health sensors to be combined into logical sensors. This allows building sophisticated multi sensor triggers.

For example, raising a safety alarm whenever a forbidden combination of I/O ports is detected.

More sophistication is brought by the use of the **<repeatCount>** property in logical sensors. For example, a repeated high CPU temperature with a low CPU load is a good detection of a failure in the cooling subsystem (failing fan, or heatsink misplacement).

The examples of XML sensor definition files provided with kehm contain some examples of logical sensors. Can you imagine some use of the logical sensors ?

3.3.2 XML Health Sensor Data Transformation

XML is the ideal format for machine to machine interfacing. Using techniques such as XSLT and XPATH, kehm output files can be filtered and transformed in almost any format.

The sample dashboards use this technique to process the XML results into html. Feel free to create your own by experimenting with the **kehm-pretty.xsl** and **kehm-devel.xsl** - section 4.7 page 19 (this can be done with the eval version!).

Source of information on XML, XLT and XPATH can be found on line. Our favorite: www.w3schools.com

3.3.3 Connecting to other Tools

Once you master a bit of XSL and XPATH, you can prepare kehm data to feed large monitoring tools designed for the cloud, or legacy on premise control and monitoring software. The idea being that kehm provides the measurements in its own data model, and the transformation techniques change this into whatever format is needed.

Some business intelligence tools can also take XML format directly, such as this example of BI Desktop visualization of kehm data.

Figure 6: Visualization of XML Monitoring Data using BI Desktop (Microsoft)

3.3.4 CMON-Line Monitoring Eval and Development Versions

CMON-Line Monitoring comes in two versions, a free evaluation and a development version. The installation process is the same and can be done in any order (e.g. installing the full version on top of the eval version or reverting to eval on a platform where the full version was installed)

- Eval version: the eval version is normally distributed along the Linux BSP and can also be obtained for free from support.KFR@kontron.com (State your platform type, Linux distribution and BSP ID)

  All examples (provided in source code and precompiled binaries) can be run, Continuous Monitoring service launched and the dashboards can be visualized and cutomized. However the eval version of the KEHM library will only accept unmodified sensor XML files (the examples delivered with the package). Modified sensor files will be rejected by the library.

- Development version: to extend or modify the initial sensor definition file, the development version of the library is needed. This is what you need to cover your complete system and application and be ready to deliver to your customer (Contact us, if you want to protect your own files for further modification by your customers).

4 / Manual Pages

4.1 kehm

NAME

kehm - extensible computer health monitoring framework

DESCRIPTION

kehm is a versatile computer monitoring technology based on user defined health sensors (see **kehm-sensors**(5)) Its main component is a library linked to the operational application or a simpler computer monitoring program. kehm computes a synthetic computer health value built from a tree of sensors. Leveraging OS commands, kehm sensors can access any information available to the main CPU. As such, the test perimeter can extend to the infrastructure outside the computer itself when the OS command tests a communication link, or a partner computer. kehm files are human readable and machine ready thanks to the use of XML format. kehm can fulfill many use cases:

CBIT

Calling kehm in a loop provides a periodic assessment of the computer health using all the defined health sensors. To limit CPU usage, the kehm library observes the refresh period of each sensor. A result file representing of the current health sensor status can be produced any time from a running program with a single line of C. It is the way to provide real-time data to external reports, views and dashboards outside the application while it keeps running. See kehm_example(1)

ACCEPTANCE

Using health sensors to cover requirements, a simple program such as kehm_example(1) can provide a synthetic or detailed assessment of the DUT (device under test) in the form of Alarm, Warning and Success count. The kehm sensor output contains enough data to use directly in test reports. See kehm-sensors(5) for details. Using a periodic dump of the data can also provide intermediate information from long test runs spanning days or weeks.

SUPPORT

kehm provides an efficient method to obtain a precise computer status from non expert operators with sensor output files. This method is compatible with the most secure application context:

- Since the files are human readable, they can be screened for sensitive data and sanitized by the end user
- Since kehm results are based on files, this support method also works for computers not connected to the outside world.
- Support teams can add more sensors to refine the information gathering from the target computer in a repeatable way.

COMPLEX SUPPLY CHAIN

kehm is extensible. Sensor definition files can leverage the knowledge of each actor involved in the final computer. Computer vendor, application integrator, end user; each can augment the existing sensor files. The final sensor collection can evaluate: the initial computer, the hardware and software modifications made by integrators and even the last changes made at commissioning time (eg IP address).

SERVICES

kehm library offers the following services API. See kehm_example.c source file for an example on how to use the library.

KEHM INIT SERVICE

int **kehm_open**(const char* sensorFile);

sensorFile is the platform XML input file. kehm_open will return EXIT_SUCCESS if the kehm initialisation is correct or EXIT_FAILURE if an error occurs such as file not found, file syntax error or xml schema error. No new initialisation can be done before **kehm_close**() is called.

KEHM LOOP TIME SERVICE

int **kehm_loop_time**(int* minLoopTime);

kehm_loop_time provides with minLoopTime the recommended period (in seconds) to use between health evaluation in the main program loop. minLoopTime is the min of all the refreshPeriod properties found in sensor definitions provided in the XML file used for **kehm_open**() . Its minimal value is 1 for one second.

KEHM EVAL SENSORS SERVICE

int **kehm_get_health_tree**(xmlDocPtr *xmlDocSensorsTree);

This service evaluates all nodes in the health tree starting with the sensors, and updates the health status and other run time properties. The returned pointer is the root node of the health sensor tree and is of type xmlDocPtr. As such it can be processed with all the standard libxml2 library services. kehm_example.exe uses this library to create a periodic XML results file on the fly while the program keeps running and will continue updating health sensor objects in memory. This XML file can then be stored or displayed using a Web browser and XSL shema file presentation (see kehm-pretty and kehm-devel for details). kehm_get_health_tree returns EXIT_SUCCESS (or EXIT_FAILURE if the library as not been initialised with a previous call to **kehm_open**()).

KEHM PLATFORM HEALTH SERVICE

int **kehm_get_health_status**(char* healthStatus);

This service provides the health status of the root node of the health sensors tree, which represents the complete platform. healthStatus can be one of ALARM/WARNING/SUCCESS. kehm_get_health_status returns EXIT_SUCCESS (or EXIT_FAILURE if the library as not been initialised with a previous call to **kehm_open**()).

KEHM TIMEOUT SERVICE

int **kehm_get_timeout_sensors**(int* timeout); int **kehm_set_timeout_sensors**(int timeout);

These services get and set the timeout value used for sensors evaluation. To protect the calling program execution flow from being blocked by a long sensor reading command execution, the library interrupts the command after timeout seconds. In addition to this, the library is equipped to adapt to transient situations (since some utilities may take a long setup time on the first call after boot). The library marks the sensor validity property as TIMEOUT_ONCE at the first timeout occurrence. It will only retry the sensor reading command on the next tree traversal. Upon a second timeout occurrence, the sensor validity property is set to TIMEOUT with a health value of ALARM and this sensor is ignored for the rest of the session.

KEHM CLOSE SERVICE

int **kehm_close**();

This service closes the kehm librarie and release its resources. It must be called if the KEHMfile sensor XML input file has been modified and must be reloaded.

DOCUMENTATION

kehm-sensors(5) the health sensor data model, found in kehm input and output files. **kehm-pretty**(5) the XML schema helpful for the results presentation. **kehm-devel**(5) the XML schema for a simplified results presentation.

FILES

kehm_example(1)

multipurpose kehm monitoring example program

kehm_walktree(1)

kehm result files post processing example

kehmcbit.service(5)

systemctl service to implement a CBIT using kehm

kehmhttp.service(5)

systemctl service to serve kehm result files to the outside world

/usr/lib64/libkehm.so

kehm development library.

/usr/lib64/libkehm-eval.so

kehm library for evaluation purpose, can be used to build any example, but only accepts Kontron pre-defined sensor input files

EVAL PACKAGE

kehm-eval(3) contains an evaluation version of the technology. It is distributed in selected linux BSP for some Kontron products such as VME and VPX blades and TRACe box for now.

OTHER

More information on computer monitoring can be found in www.kontron.com/ technologies/ in CMON-line and FLEET-Line data sheets.

SEE ALSO

kehm-eval(3), kehm-devel(5), kehm-sensors(5), kehmcbit.service(5), kehmhttp.service(5)

4.2 kehm-sensors

NAME

kehm-sensors - data model for extensible computer health monitoring framework

DESCRIPTION

kehm relies on health sensors to monitor a computer status. The data model is designed to be used both in input and output XML files. The kehm library uses the input file to create in-memory objects as a tree of nodes, and manages them during the whole execution of the main program. Each time the computer health needs to be assessed, the tree is traversed, and nodes are evaluated and updated. In-memory nodes can be dumped at any time into a XML output file. Many kehm nodes and attributes are informational and not used by the kehm library itself. Their purpose is to help in post processing the output for dashboards, reports and analytics.

XML Schema

The global structure of a kehm XML file is:

<kehm><sensorGroup><sensor></sensor></sensorGroup><logicalSensor><logicalRule></logicalRule></logicalSensor></kehm>.

The XML Schema is located in /usr/include/kehm/kehm.xsd.

Input XML files which do not follow this XML schema will cause an error during file parsing.

NODES

<kehm>

is the top level node. Its platform attribute is not used by the kehm library, but useful for reports and dashboards. This top level node contains:

<sensorGroup>

sensorGroups are used to group sensors and compute a health value for the group. kehm maintains the value of <health> and <lastRefresh> at the group level. Name and Desc attributes are not used by the kehm library. There can be as many sensorGroups as necessary.

<sensor>

is the most important object. The content of a sensor is described in the SENSOR NODE section below.

</sensor>

</sensorGroup>

<logicalSensor>

is an optional node of the top level. It contains only logicalRules nodes built from a combination of <sensor> defined above.

<logicalRule>

nodes are built from existing sensors. They refer to and are used to combine different sensor in a AND combination (eg if the CPU load is less than a value AND the CPU temp is over a temp value). Sensors are referenced using their name. This is why the name node of a sensor node must be unique.

</logicalRule>

</logicalSensor>

</kehm>

SENSOR NODE

<sensor>

the sensor node supports a desc attributes which contains a small description text. Desc is not used by kehm library.

<name>

sensor name. Name must be unique since it is used as the sensor ID in <logicalRule> nodes.

<class>

informational only, may be used for subgrouping sensors in post processing. Not used by the kehm library

<domain>

domain has been introduced to allow specialized views in dashboards. **kehm-pretty.xsl** uses this in its domain tab. This node can contain multiple words, allowing a sensor to appear in more than one view. Not used by the kehm library.

<value>

contains the last sensor reading (if validity is VALID)

<health>

contains the health value of the sensor (Alarm, Warning, Success) computed from the <alarm> and <warn> rules and the sensor reading. (if validity is VALID)

<validity>

VALID/INVALID(errormsg)/**TIMEOUT_ONCE/TIMEOUT**, value and health nodes of a sensor can be used whenever this node contains "VALID". This node can also be used to report errors encountered when evaluating the sensor using the <systemCall> and the <serviceFamily>.

<action>

is the place for a recommendation of what to do in case of warning or alarm. Not used by the kehm library.

<visible>

true/false. Informational, could be used to filter out in reports. Not used by the kehm library

<handler>

Where the information is coming from. Usually CPU except when a specific monitoring co-processor is used (eg HMU in TRACe products). Informational, Not used by the kehm library

<log>false</log>

true/false. informational, could be used by post processing stages filters to limit the size of log files. Not used by the kehm library.

<alarm><warn>

are trigger nodes for the sensor. They define how a sensor reading should be evaluated into a health value using their subnodes as such:

<repeatCondition>

contains a repeat count required to validate the trigger.

<repeatOccurence>

is used by the library to maintain the current count and compare it with <repeatCondition>

<rule> an <alarm> or <warn> entry can contain multiple <rules>. A <rule> gives the condition that will raise an ALARM or WARN state for the sensor. If only one condition is meet the sensors state will be set to ALARM or WARN, such as in multiple entry OR statements. A rule boolean value is assessed by the library using <operator> [equal|notequal|greater|lesser] and <operandValue> pairs to compare with the sensor reading.

<maxValue><minValue>

are maintained by the library during the process execution. This information is available in kehm XML output files and is an efficient way to capture new sensor boundaries during long test sessions and set the triggers for Alarms or Warning accordingly.

<alarmCount><warnCount><callCount>

the library maintains this counts to help compute alarm and warning rates. <callCount> also help understand <refreshPeriod> settings

<position>

Not used by the kehm library, could be used to plot 3D models using the position of the sensor in the computer, a useful information for temperature sensors or user interface connectors and LEDs.

<refreshPeriod>

the time in seconds between sensor readings. The main program needs to call into the kehm library often enough to evaluate fast moving sensors. However, it would be a waste of time to also read all slow moving sensors at the same cadence. <refreshPeriod> is the time in seconds to wait between subsequent readings of the sensor. A value of -1 means service to be called only at first evaluation. A value of 0 means service to be call any time the sensors tree is evaluated.

<lastRefresh>

a timestamp of the last sensor reading.

<systemCall>

All the necessary command parameters needed to extract the reading of the sensor. Depends on the <serviceFamily>

<serviceFamily>

Is used to select the command mecanism used to read the sensor. Can contain [system|KEAPI|USB-HMU-HID|USB-DHMU-HID]

system

Creates a subshell with /bin/bash to execute the cmd string found in **<systemCall>** and stores its std output (fd=1) in the sensor **<value>** node.

Errors:

If the subshell takes more than timeout seconds to exit, the sensor value is set to nothing and **<validity>** set to TIMEOUT_ONCE. The validity is set to TIMEOUT if the previous call was set to TIMEOUT_ONCE so that the call in timeout is not be called again.

If the cmd string fails then the sensors **<state>** is set to INVALID "(error msg)". "error msg" is made from the subshell stderr (fd=2). If empty, the error code returned by the subshell is used instead. A return code 1 is not considered as an error by libkehm.

The subshell invokes /bin/bash with *-euo pipefail* options. This leads to the following behavior: (-e option) subshell exits immediately if cmd string (a pipeline, a simple command, a list of commands, or a compound command) exits with a non-zero status at first error.

(-u option) It treats unset variables as an error when performing parameter expansion (except for the special parameters "@" and "*")

(-o pipefail) The return value of a pipeline is the value of the last (rightmost) command to exit with a non-zero status

KEAPI

calls into the kontron KEAPI HAL abstraction library which exist on many OSes. Example of <systemCall> for KEAPI : KEApiGetBoardInfo(lastRepairDate) .

More on KEAPI in /usr/share/doc/keapi

USB-HMU-HID

uses the USB command interface of the Health Management Unit found in TRACe transportation box computers.

USB-DHMU-HID

uses the USB command interface of the Display Health Management Unit found in TRACe-D transportation HMIs.

<returnType>

can be [string|integer] to help with the comparison operators inside the library. If [integer] the comparison is done between integers after a atof() conversion of <valueString> and <operandValue>. If <returnType> is [string] the library uses strverscmp(3) which is an elaborate C library routine to compare strings made of alphanumeric and numeric characters.

<infoText>

is the place for a longer text explaining the goal of the sensor

LOGICALRULE NODE

<name> Logical Rule Short name

<class> same as SENSOR NODE, will be set to logicalSensor here

<domain> same as SENSOR NODE

<health> same as SENSOR NODE

<validity> same as SENSOR NODE

<action> same as SENSOR NODE

<visible> same as SENSOR NODE

<log> same as SENSOR NODE

<alarm> <warn> same as SENSOR NODE

<repeatCondition> same as SENSOR NODE

<repeatOccurence> same as SENSOR NODE

<alarm> or **<warn>** entries can contain multiple **<sensorID>**. A <sensorID> gets the value of the sensor identified by its unique name <name> and introduces the conditions that will raise an ALARM or WARN state for the Logical sensor. All given condition must be meet to set ALARM or WARN state, such as in multiple entries AND statements. A boolean rule value is assessed by the library using <operator> [equal|notequal|greater|lesser] and <operandValue> pairs to compare with the sensor reading.

FILES

/usr/lib64/libkehm-eval.so

kehm library for evaluation purpose, can be used to build any example, but only accepts kontron pre-defined sensor input files

/usr/lib64/libkehm.so

kehm development library.

[platform]-sensors.xml

default health sensor input file. Supported [platform] are TRACe, VM60Sx, VX3058. See kehm_example for details. kehm-RESULT.xml default health sensor output file generated by kehm_example. See kehm_example for details.

kehmcbit.service(5)

use this to run kehm_example as a CBIT service under systemctl.

ENVIRONMENT

Execution errors go to stderr, stdout is used for health status

BUGS

please contact support-kfr@kontron.com

AUTHOR

vincent.chuffart@kontron.com

EVAL PACKAGE

kehm-eval(3) contains an evaluation version of the technology. It is available in selected BSP of some Kontron products such as VME and VPX blades and TRACe box for now.

OTHER

More information on computer monitoring can be found in www.kontron.com/technologies/ in CMON-line and FLEET-Line data sheets.

SEE ALSO

kehm(3), **kehm-eval**(3), **kehm-devel**(5), **kehm_example**(1) , **kehmcbit.service**(5), **kehmhttp.service**(5)

4.3 kehm_example

NAME

kehm_example - multipurpose computer health monitoring example

SYNOPSIS

kehm_example.exe [**-i** sensor-definition-file.xml] [**-o**[sensor-status-file.xml]] [-c|a|w|s]

DESCRIPTION

This kehm sample program takes a xml input file of sensor definitions, evaluates them and provides a summary of the platform health. On demand (**-d**) it creates an output file similar to the input, augmented with the sensor measures and health status. Using one of the infinite loop severity options, it can be used as a continuous monitoring engine, which reports a digest of the sensor health on the standard output. This program also re creates the xml output file at each loop, allowing other programs to build real time reports and dashboards.

OPTIONS

-i

input sensor definition file, defaults to kehm.xml (create a link with ln -s target_file.xml kehm.xml if necessary) -o[output sensor definition + value file], defaults to kehm-RESULT.xml. Can be used by other programs, eg http service. No result file is created if -o is not specified. Because it is optional the argument must immediately follow -o without any blank space

-c|a|w|s

infinite loop, choose one option only

-c(ontinuous)

loops, print platform top level health status

-a(larm)

loops, print top level plus sensors with health = alarm

-w(arning)

loops, print top level plus sensors with health = warning or alarm

-s(uccess)

loops, print top level plus all sensors health results

EXAMPLE

kehm_example.exe -i kehm.xml -o -c

This command takes kehm.xml as input sensor definition list file (option -i), run continuously and print the Global sensor tree health (option -c) and output a result file at each loop into default file kehm-RESULT.xml (option -o without argument)

FILES

/usr/lib64/libkehm-eval.so

kehm library for evaluation purpose, can be used to build any example, but only accepts kontron pre-defined sensor input files

/usr/lib64/libkehm.so

kehm development library.

kehm.xml

default health sensor input file. See kehm-sensors(5) for details.

kehm-RESULT.xml

default health sensor output file. See kehm-sensors(5) for details.

kehmcbit.service(5)

use this to run kehm_example as a CBIT service under systemctl.

ENVIRONMENT

Execution errors go to stderr, stdout is used for health status

BUGS

please contact support-kfr@kontron.com

AUTHOR

vincent.chuffart@kontron.com

SEE ALSO

kehm-eval(3), kehm-devel(3), **kehm-sensors**(5), **kehmcbit.service**(5), **kehmhttp.service**(5)

4.4 kehm_walktree

NAME

kehm_walktree - simple computer health monitoring results display example

SYNOPSIS

kehm_walktree.exe [**-d** n] -s [alarm|warning|success] [**-f** result.xml }

DESCRIPTION

This kehm sample program takes a xml health output file, which contains health sensor values and provides a summary of the platform health to a given depth, and a given severity level. In this particular example, nice colors are added to the output around the words ALARM (red), WARNING (orange) and SUCCESS (green) using special VT escape sequences. The look may vary depending on the terminal program used.

OPTIONS

-d

depth level with n=1 or 2 or 3. Only walk health sensor tree down to given depth. n=1 give only top level health status (see kehm-sensors(5) for kehm XML node structure)

-f

sensor health results file to use, defaults to kehm-RESULTS.xml

-s [alarm|warning|success]

severity filter. Use one of:

A(larm)

print top level plus sensors with health = alarm

W(arning)

loops, print top level plus sensors with health = warning or alarm

S(uccess)

loops, print top level plus all sensors health results

FILES

kehm-RESULT.xml

default health sensor output file. See **kehm-sensors**(5) for details.

EXAMPLE

kehm_walktree.exe **-d** 3 -s success **-f** kehm-RESULT.xml

Display the health (ALARM/WARNING/SUCCESS) of the full sensor tree (level 3 of depth) with states in color

ENVIRONMENT

Execution errors go to stderr, stdout is used for health status

BUGS

please contact support-kfr@kontron.com

AUTHOR

vincent.chuffart@kontron.com

SEE ALSO

kehm-eval(3), kehm(3), **kehm-sensors**(5)

4.5 kehmcbit.service

NAME

kehmcbit.service - simple CBIT example using computer health monitoring framework

DESCRIPTION

This file describers a service using **kehm_example**(1) as a Continuous Built In Test (CBIT). In this use case, the program reads the sensor file in and loops continuously at the cadence of the fastest sensor refresh period (see **kehm-sensors**(5) for more on <refreshPeriod>).

The kehm_example is used here to output sensors in Alarm state on stderr and also produce a complete sensor result file. This is done at each loop

To use this service, kehmcbit.service file needs to be copied to /etc/systemd/system and started using "systemctl start kehmcbit" You may also enable it for further boot with "systemctl enable kehmcbit".

The service status can be verified using "systemctl status kehmcbit" and all the messages of the running program appear in the system journal. kehmcbit messages can then be dislayed with "journalctl _SYSTEMD_UNIT=kehmcbit.service"

TODO

It is a good idea for continuous use to configure the maximum size allocated to the system journal using SystemMaxUse parameter in /etc/systemd/journald.conf to prevent file system free space exhaustion.

SEE ALSO

kehm_example(1), **kehmhttp.service**(5), **systemd**(8), **systemctl**(1), **journalctl**(1), **journald.conf**(5)

4.6 kehmhttp.service

NAME

kehmhttp.service – http service for computer health monitoring XML files

DESCRIPTION

This **systemd**(8) service definition file provides a simple http service useful to display kehm XML result files with a http browser. In this use case, the python SimpleHTTPServer module is invoked as a daemon running in /usr/share/kehm

BENEFITS

Opening kehm XML result files with a http browser allows:

- to view the health information using XSL style sheets provided with kehm, or defined by the end user.
- to view the system health information from another computer.

For this to work, the firewall must authorize the use of port 8000. For example to allow TCP traffic to port 8000, enter the following command as root:

```
# firewall-cmd --add-port=8000/tcp
```

To make this setting permanent, add the **--permanent** option and reload the firewall.

USAGE

To use this service, the kehm user must be created, kehmhttp.service file needs to be copied to /etc/systemd/system and started using "systemctl start kehmhttp" When the service is enabled with "systemctl enable kehmhttp", it will be restarted after a reboot.

The service status can be verified using "systemctl status kehmhttp" and all the messages of the running program appear in the system journal. kehmhttp messages can then be displayed with "journalctl _SYSTEMD_UNIT=kehmhttp.service". With the current settings they give the remote client IP and the file served to it.

SEE ALSO

python(1), **systemd**(8), **systemctl**(1)

4.7 kehm-pretty.xsl kehm-devel.xsl

NAME

kehm-pretty.xsl, kehm-devel.xsl - style sheets for computer health monitoring XML files

DESCRIPTION

The combination of style sheets with XML allow to create elegant views and feature rich dashboards out of xml files. Style sheet contain style clauses and XPATH directives, allowing filtering, counting, selective display etc. Opening a kehm XML result files with a http browser is the easiest way to display the final result.

XSL files are installed into /usr/share/kehm

USAGE

Start from the provided examples and tweak them to create the view you need.

kehm-devel

provides a one page HTTP document with the top level health, the sensor groups, and the individual sensors details.

kehm-pretty

leverages the BOOTSTRAP (https://en.wikipedia.org/wiki/Bootstrap) HTTP framework to provide an elegant , multi view dashboard from the same XML health sensor results files.

By default kehm.xsl in invoked from the platform-sensors.xml file. The simplest is to create a symbolic link towards the wanted XSL file For example:

For pretty presentation

```
ln -s kehm-pretty.xsl kehm.xsl
```

For development

```
ln -s kehm-devel.xsl kehm.xsl
```

TODO

Since **kehm-pretty** relies on the BOOTSTRAP framework, the http browser should have access to the Internet to load the framework files dynamically. If you cannot connect to the outside world, then use a local copy of BOOTSTRAP (see http://getbootstrap.com/getting-started/#download)

SEE ALSO

kehmhttp.service(5) https://www.w3schools.com/xml/ is a good starting point for reference information on XML, XPATH, XSD.

5 / Known Issues

5.1 RAM Consumption with Firefox and kehm-devel.xsl

When viewing `kehm-result.xml` formatted by `kehm-devel.xsl` in a web browser, the page is automatically reloaded every 20 s. We have observed the following issue: Firefox allocates more and more data space at each page reload, leading to free mem consumption on the system running the Firefox browser process. This memory is only released when all Firefox windows are closed.

We have not observed this issue with Internet Explorer or Chrome.

About Kontron - An S&T Company

Kontron is a global leader in IoT/Embedded Computing Technology (ECT). As a part of technology group S&T, Kontron offers a combined portfolio of secure hardware, middleware and services for Internet of Things (IoT) and Industry 4.0 applications. With its standard products and tailor-made solutions based on highly reliable state-of-the-art embedded technologies, Kontron provides secure and innovative applications for a variety of industries. As a result, customers benefit from accelerated time-to-market, reduced total cost of ownership, product longevity and the best fully integrated applications overall.

For more information, please visit: **www.kontron.com**

CORPORATE OFFICES

FRANCE

150, rue Marcelin Berthelot
ZI de Toulon-Est - BP 244
83078 Toulon Cedex 9 - France
Tel: +33 4 98 16 34 00
Fax: +33 4 98 16 34 01
sales.KFR@kontron.com

GLOBAL HEADQUARTERS

Lise-Meitner-Str. 3-5
86156 Augsburg
Germany
Tel.: + 49 821 4086-0
Fax: + 49 821 4086-111
info@kontron.com

NORTH AMERICA

9477 Waples Street, Suite 150
San Diego, CA 92121
USA
Tel.: + 1 888 294 4558
Fax: + 1 858 677 0898
info@us.kontron.com

ASIA PACIFIC

1~2F, 10 Bldg, Nº 8 Liangshuihe 2nd Str.
Economical & Techno. Develop. Zone,
Beijing, 100176, P.R. China
Tel.: + 86 10 63751188
Fax: + 86 10 83682438
info@kontron.cn