Integrated Server Monitoring

System for automated supervision of computer servers

The Next Generation Monitoring

© Dipl.-Inform. Wilhelm Buchholz

http://www.monitor-site.de

The system offers the following functions:

- Autonomous agents for Linux, Windows, AIX, Solaris, HP-UX, Mac OS X
- Different forms of the log file analysis, full integration of journalctl (systemd)
- Monitor scripts and correction scripts
- Filtering and display of SNMP-Traps
- Active monitoring of remote tcp ports
- Central Management Station with graphic (X11) control surface und web surface
- Multi-user ability
- Integrated data retention with dynamic store management and ring store
- Parallelism by multithreading, asynchronous processing
- Process display and status display at the Management Station
- Dynamic registration and heartbeat
- Filter functions to the difference display
- Forwarding of messages, automatic actions
- Central administration of the configuration files and administrative access to the nodes
- Encrypted communication with the agents over one port (tcp/udp)
- Both ipv4 as well as ipv6
- Program completely in C++
- Management Station for Linux 64 Bit
- Comfortable operation

This system, developed from many years of operating experience, is manageable, easy to install and operate. It has a bandwidth of a few dozen to several thousand servers (nodes), is universally applicable and thus offering the best conditions for a wide audience. Due to the multi-user capability it is also suitable for a large, enterprise's division of labor.

Contents

1. Introduction

The term monitoring refers to the timely identification of relevant customer problems on computer servers from a central control room, thus increasing the availability of computer systems. The present system for real-time monitoring is a counter project to the commercial frameworks of well-known providers developed in nineteen-nineties.

These systems, you cannot even install without training or foreign help, have a bloated complexity that is in no proportion to the operational requirements also having a negative impact on the cost of networking. The cost of acquisition, operation, learning curve, "consultants" can only be described as adventurous. The use of such tools per se is a not-insignificant problem.

Parallelly - also in response to the difficulties with the large commercial systems - a number of smaller tools (including open source) have emerged, which may be a good low-price alternative with regard to purchase and operation costs, but lacking functionality. This applies, for example, to a practical, content log file analysis, for which visual effects are less important than gaining more leading (proactive) information about a server. It also refers to a reasonable presentation of results combined with persistent data storage. The capacity problem, i.e. the capacity of **one** central Management Station for a certain number of clients is usually not solved. The system depends on all monitored servers not just on one.

The presently existing systems tend to shift problems the provider should actually solve on to the user, who is suddenly confronted with development and/or design problems. Intended benefits had the reverse effect (for large systems development and test environments are mandatory). The risk of having to deal with a permanent "building site" with no actual profit is great. The operating cost is far too high (which - by the way – contradicts the basic idea of automation).

This system takes into account that today the tasks in server monitoring are largely known. This includes the automatic and immediate analysis of a series of log files (even for individual applications) on a server, transferring lines revealed by search patterns representing them centrally. In addition, there are monitoring scripts and standard monitoring thus bundling the most important, problem-related monitoring functions. Further functions for escalation of fault messages and for sending e-mails are also integrated.

Furthermore, it is considered that the number of servers has increased sharply since the early days and that there are network restrictions such as firewalls, (double) address translation, ipv6 and new modes (cloud computing, grid computing). Slow network connections provide no obstacle. For data transfer only one port tcp/udp is required. Through a dynamic encryption (AES/CBC, MRC4), the Internet may be used as a transmission medium. As a central output, there is a process display for error messages and a status display for all servers contained in the monitoring, each as a graphical user interface and a web interface. The process display is able to represent and to document a failure as a dynamic process with a beginning, end and duration.

The system has defined functions, so that it is not necessary to use any kind of "plugins" or "smart plugins".

2. Agents and result display

The focus of monitoring is on the agents, which belong to the system.

Agents are programs that run on the servers to be monitored (= nodes) and periodically perform queries. The result is sent in form of an event (= message) via the network to a central Management Station where it is visibly displayed. The events will be encrypted with a secret key and a session key (session keys mean that every message is encrypted differently). In addition an integrity check by checksums takes place.

The transfer to the Management Station is performed in one direction via an arbitrary tcp port. Each agent is parameterized using a configuration file, integrating the parameters to be monitored and the port and address of the Management Station. This can also be a numeric IP address, because the communication takes place independently of DNS (domain name service). A name resolution on the Management Station is not necessary.

Optionally, the address of an alternate Management Station can also be specified in case the first one fails.

The use of autonomous agents has the following advantages:

- Low network load: It is reported only in case of failure. You have to realize that well over 90% of the local queries run on the node are negative (which is desirable) and the network is not used.
- Relief Management Station queries. As the number of servers to be monitored increases the load increases linearly rather than progressively.
- Security: Messages are sent; they cannot be queried from the outside. Due to the agents, the nodes are not even potentially vulnerable.
- No loss of information at (short-term) network faults. The agents save the messages locally subsequently delivering them when the fault ends.

The use of agents is - with regard to the resource load - much cheaper than the continuous queries over the network from a Management Station. The network connection to the Management Station is opened and closed as needed. There is no permanent tcp connection.

The agents can be used either as a background process or as a batch program (exception: “asyncmonagent”, see below). For background processes

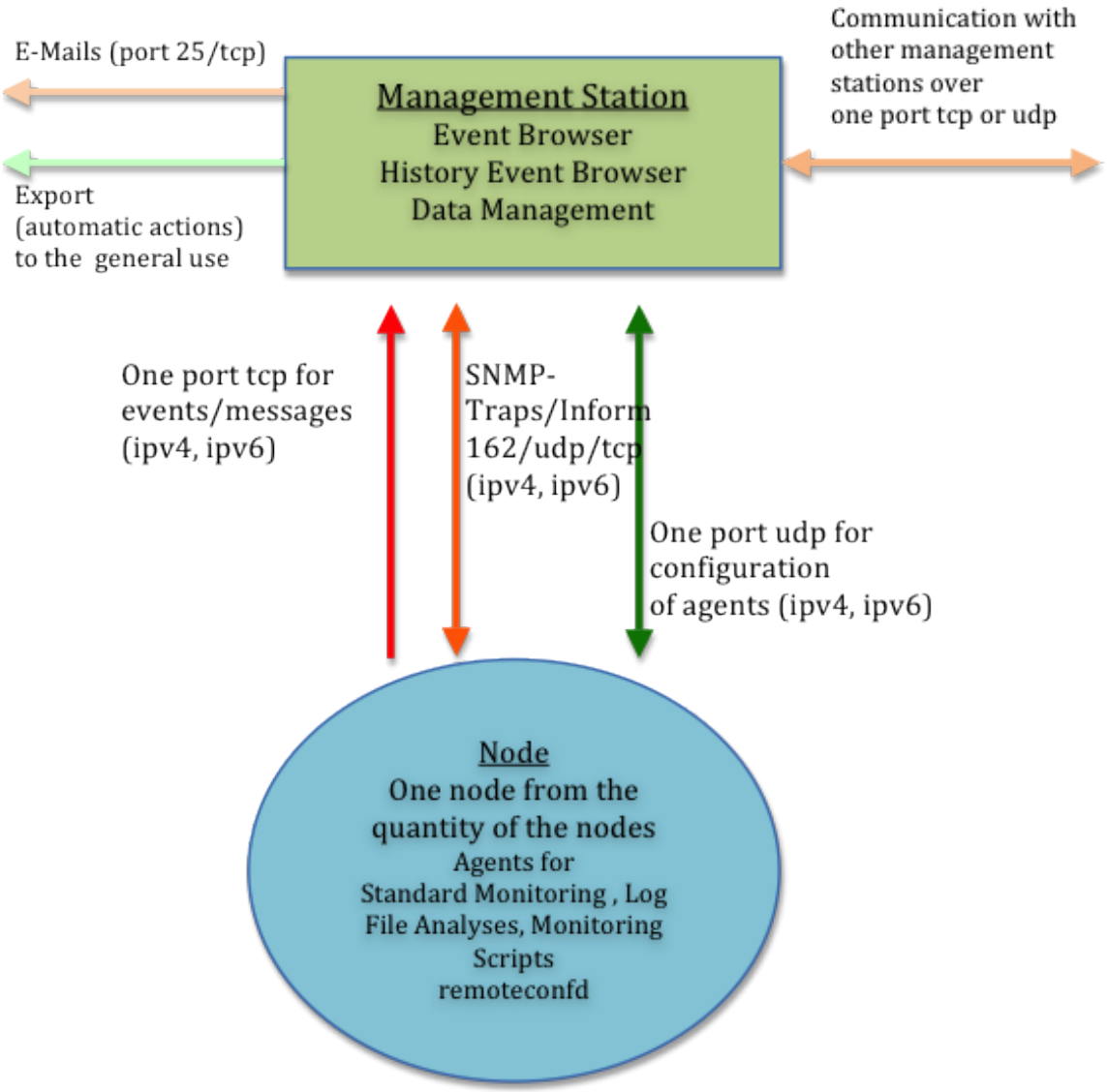
the polling interval is fixed in the configuration file by the time unit seconds (e.g. 300 for five minutes). In case of a subsequent modification the configuration file will be automatically read and the event is signaled to the Management Station.

Without the entry for the polling interval, the agent terminates after each call. Afterwards, it is periodically called by a local scheduler (for example crontab for Unix, Task Scheduler for Windows). The agents do not need root or administrator privileges.

On the Management Station, the events will appear in chronological order as a process display in the "Event Browser", which is available either as a X11 program or as a web application. In contrast to a pure status display the messages are preserved, not getting lost by overwriting them with a new status. This means that faults can be identified as a dynamic process making it possible to trace them back subsequently. Furthermore, multiple faults can also be represented. For example, more than just one file system can fill up, which is indicated by individual messages (for large Unix servers dozens of file systems are possible). The same is true for the ample field of log file analysis, where the encountered lines of the log file are represented as an event text.

The system uses the character sets UTF-8 (Unicode), ISO-8859-1 to ISO-8859-10, ISO-8859-13 to ISO-8859-16 and CP1250 to CP1258 (Windows).

Data Flows



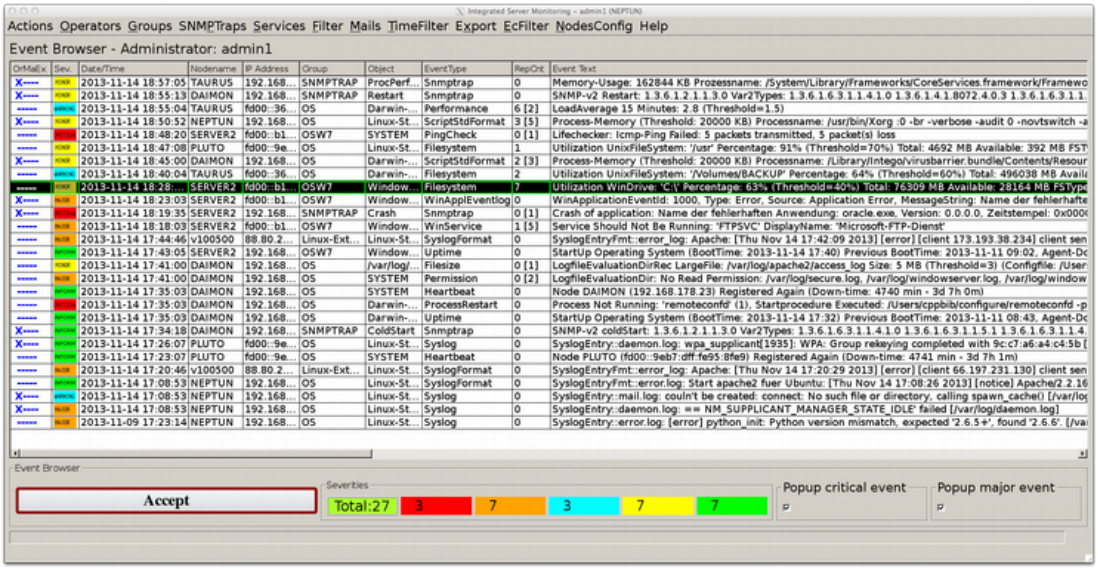
The diagram shows the communication of agents with the Management Station. The configuration files for the agents are text files that can be edited by a local editor.

There are five different severities for the system:

1. inform
2. minor
3. warning
4. major
5. critical

A message consists of the following attributes: severity (color-coded), receiving time (ISO date format), node name as the network name of the message source (string constant), IP address (ipv4 or ipv6), group (string constant), object (string constant), Event Type (string constant), RepCnt (number) as a repeat count, a counter for oppressed messages of the same type and finally the event text that may have a length with up to 1024 bytes. Furthermore, there are several hidden attributes, such as the creation time on the node (UTC time stamp), the name of the operator who accepted the event and the time of the acknowledgement. The attribute "Group" (message group) determines the assignment to the operators.

X11-User Surface (process display)



The figure shows the Event Browser of the central Management Station with a sequence of current messages. Each row represents an event or message. The most recent message appears at the top, the oldest one at the bottom. The second column on the left is the message severity (severity), followed by date and time, the last one on the far right side shows the event text, providing a unique description of the event. The message text is of outstanding importance, because in combination with the severity, it determines the basis for the respective error handling.

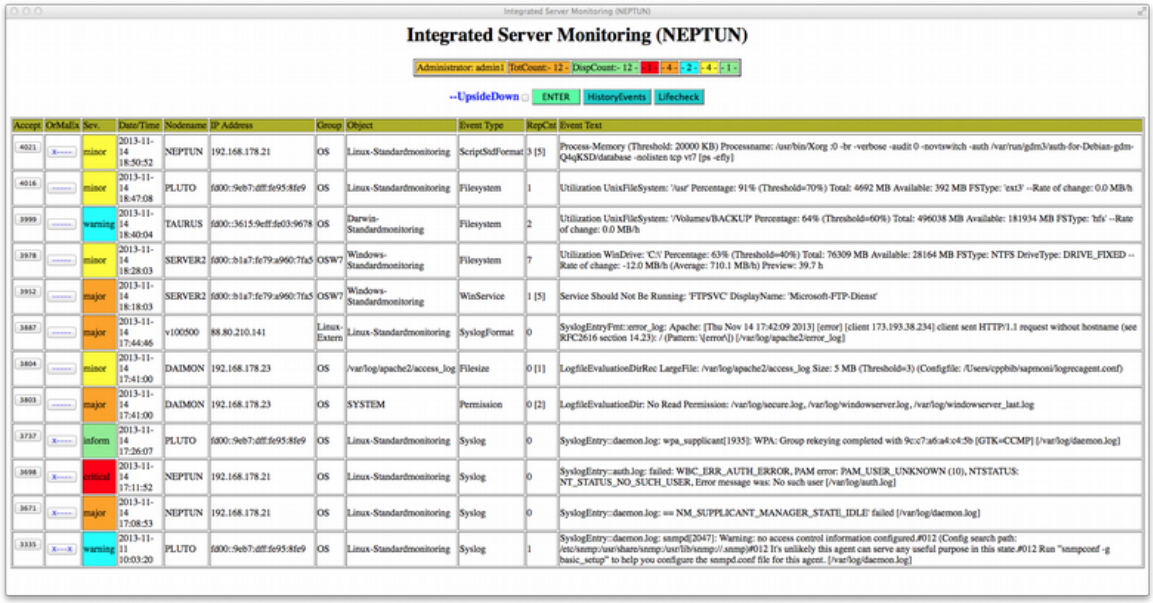
With the push button "Accept" (bottom left), a message is removed after processing, brought from the state "current" to the state "archived". It will no longer be visible to other users, but it is possible to bring it back in a different part of the user interface, the "History Event Browser".

At the top of the image there is a pull down menu which lead to the data entry screens for the various settings of the system. It differs depending on whether you log on as an administrator or operator. An administrator, of which there may be several, has the authority to make any settings and basically looks at all messages. By contrast, an operator sees only those messages whose groups he is assigned to. He is allowed to configure the corresponding nodes with the menu item "NodesConfig".

All system users must log on with user ID and password. A user is allowed to log in several times.

The optical appearance of the X11 interface (colors, font sizes, frames, etc) can be customized using style sheets.

Web Surface (process display)



The figure shows the same view as a web interface for active (non-working) messages. A message is acknowledged after processing by a push button on the left side. Afterwards, the message disappears. The number in square brackets in the column "RepCnt" indicates the number of suppressed messages according to the set filter in a certain period, so that it is possible to control the frequency.

The form of representation as a temporal sequence of events is a fundamental requirement for an operational monitoring tool.

3. Replacement Mechanism (Format Statement)

The replacement mechanism is intended for the user-friendly design of an event message text. It is used on the Management Station for configuring SNMP Traps and for the filters and with regard to agents for log file analysis and for the evaluation of monitoring scripts.

The use of the replacement mechanism allows for an incoming text, which consists of a series of words or columns, to be rearranged, shortened (i.e. unwanted parts of the text can be removed), and complemented by new information. The added information may also be troubleshooting instructions.

The mechanism is implemented by a format string containing the operators '$', '%', '&', @,? and '-', followed by a number or a substring.

This way, using the format string you can convert an incoming line into an outgoing text (= transformation instructions).

- $n or ${n}: n is a number [1..99]. Outputs the <n>th word of the input text
- %n oder %{n}: shift to the left, outputs the <n> columns to the left shifted input text, the input line remains unchanged
- &n or &{n}:Shift of <n> characters to the left of the input text, there is no immediate output, the new beginning of the text input field is automatically set to the beginning of a word or column
- &{n,m}: Outputs <m> characters from the <n>th character of the input line
- &{n[|#]substring}: Search for a sub-string in a word. Outputs from the <n>th character to the sub-string substring in the same word. If substring is not found, the output is to the end of the word
- @n or @{n}: Shift to the left by <n> columns in the input line, the original column <n+1> is then the beginning of the input line, there is no direct output
- %<[n|]substring>: Search for a sub-string in the whole line or optionally after the <n>th occurrence of a sub-string in the line (n > 0). Then shift left until substring in the input line. The sub-string found is the new beginning of the input line, there is no direct output
- ?<[n|]substring>: Outputs the sub-string shifted to the left of the text input line. If sub-string is found, the formatting terminates, otherwise it is continued with the following special characters

- -<[n|]substring>: Outputs the input text truncated at the point of occurrence of sub-string substring, the found substring is cut off, the input line remains unchanged
- $*: Outputs the whole line
- $$: Outputs the last column of the input text

The search for substrings takes place from left to right; this also applies to the format statement which is processed from left to right. The operators can be combined. For example, you can switch the first word with the second in the message text putting in between an arbitrary string. In case the operators are not given in the required formatted text, the incoming text is completely replaced by the string constant.

Example:

When evaluating the syslog file on a Linux server, the following line has been determined, which is the input text for formatting:

Oct 20 16:06:45 v100500 sshd[3992]: Address 69.65.49.82 maps to guryat.com, but this does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!

Formatstring: “Break-in attempt per ssh (affected server: $4): %5“

Output for the event text:

Break-in attempt per ssh (affected server: v10050): Address 69.65.49.82 maps to guryat.com, but this does not map back to the address - POSSIBLE BREAK-IN ATTEMPT!

The operator "%5" causes the output of the five columns/words shifted to the left input line. "$4" is the fourth word in the input text to the desired location of the source text. The formatted, actual text information can now be tested for equality, so that it would appears only once within a selectable period of time (e.g. 10 minutes) (in case of an intrusion, there may be hundreds of entries of this type within a few minutes).

If the format statement is missing, the text is in full output.

4. Regular Expressions (Search Patterns)

The system uses *extended regular expressions* according to the POSIX standard as search pattern. The properties can be found in the *manual pages* of Unix. For the special requirements of this system, there are optional additions that are appended to the end of the search pattern after a slash '/'.

The syntax is: <RegExp>[/i|v|!]

The real search pattern followed by '/' and 'i' or 'v' or '!'.

The meaning of the symbols:

- 'i': Perform case insensitive matching
- 'v': The search result is reversed, case sensitive matching is performed
- '!': The search result is reversed, case **insensitive** matching is performed

The special importance of the option can be switched off with a preceding backslash '\'.

Examples:

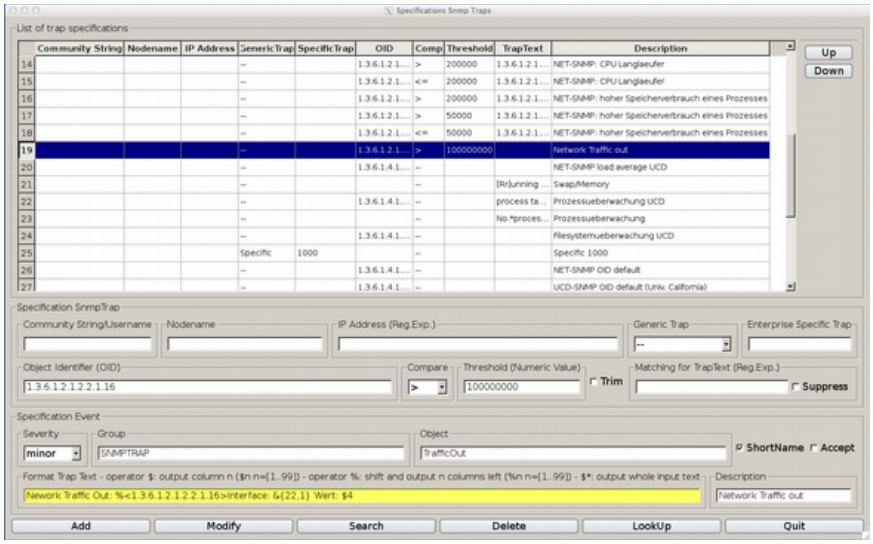
"^os$/i" will match the string "OS", "Os", "oS", "os"
"fatal/i" will match lines containing "Fatal", "FATAL", "fatal", ...
"[0-9]/v" will match lines not containing any digits
"ABC/v" will match lines **not** containing "ABC"
"ABC/!" will match lines **not** containing "Abc", ABC", "abc", ...
"[ ][1-9][0-9]{1,2}[ ]/v" will match a number that has more than three places
"[ ]3\.14[0-9]*[ ]" will match the number 3.14...
"[ ]([3-9][0-9]{5})|([1-9][0-9]{6,})[ ]" will match a number that is greater or equal 300000

5. SNMP-Traps (Trap Receiver)

SNMP (simple network management protocol) is a standard means for the system management, which is platform independent. Traps and Notifications will be autonomously sent by the servers on port 162/udp (not to be confused with port 161/udp, via which queries from the outside to a server are possible). On the servers the background process "snmpd" must be active and in the configuration file the Management Station must be determined as trap destination. For Windows there is the SNMP service.

The system is capable of receiving, filtering and displaying SNMP Traps version 1, 2c and 3 (traps of version 3 only if they are **not** encrypted, more see below). The filtering is done with an ordered list of specifications, which decides by comparing the conjunctive components on whether and how an incoming trap is shown. The received messages can also be traps of other SNMP-enabled devices such as routers or network printers.

The setting for SNMP Traps takes an administrator using the graphical user interface:



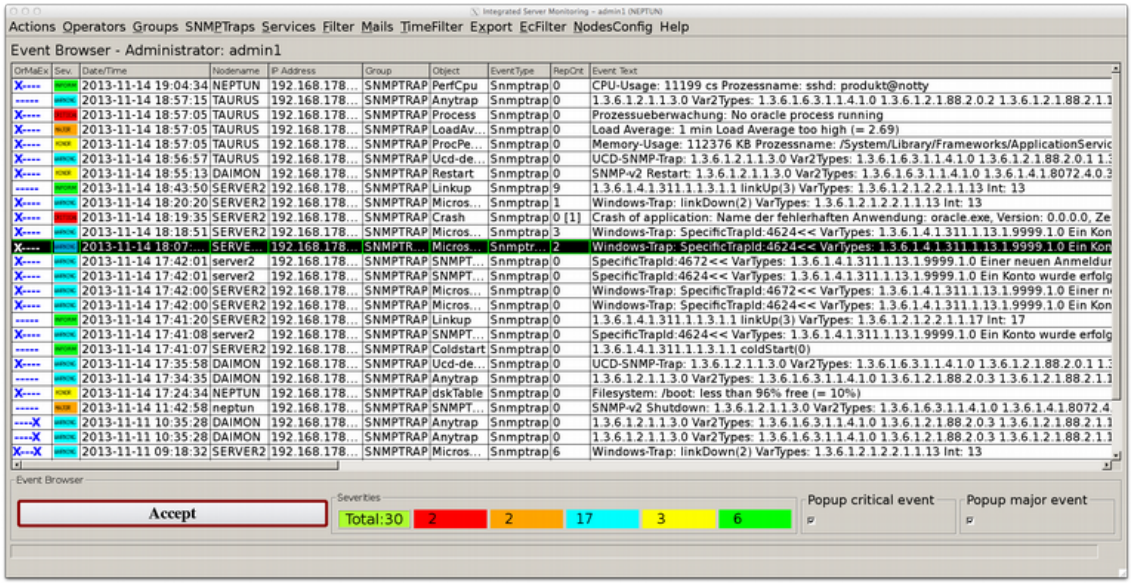
The figure shows the input mask specifications of trap messages. The upper part is the table of already existing descriptions. You can add, modify, delete and search. The length of the table is not limited.

A trap is specified by the fields "community string", "Generic Trap" (v1), "Enterprise Specific Trap" (v1), "Object Identifier (OID)" and "Matching for Trap Text". You can appeal selectively to single OID's within a trap and compare its associated numerical value with a threshold. Similarly, it is possible

to provide the same numerical value with different thresholds and severities. The fields "Node Name" and "IP Address" are provided in case there is no name resolution for an incoming IP address, or you want to give an alias name of your choice (for example routers). If a field is empty, the comparison is positive. If all fields are empty or inactive, it means: Each trap.

In the lower area you have to enter the attributes of "Severity", "Group" and "Object" for a message. The input field "Format Trap" optionally defines a format string, which converts the incoming message text into the output text.

When a trap message arrives, the table is processed line by line from top to bottom. When the specification of a line is identical with the trap message, the result is output in accordance with the agreements made. Then the process terminates. Traps can be suppressed with the checkbox "Suppress". The processing sequence can also be used to process trap types so far unknown.



The figure shows a selection of trap messages. A trap is defined as a sequence of numerical OID's and corresponding value shown (the numeric OID comes over the net and is authentic). SNMP strings ("octet strings") play a special role, serving either as a label for numeric values (for example, process name) or containing independent information. Thus, an application or a subsystem sends an entire text line as log information, which is then represented as an event text.

This system is prepared to listen to port 162/tcp (or any other port number). However, the use of tcp with SNMP is rather the exception.

SNMP-v3:

To receive encrypted traps (*secLevel*: *authPriv*), you can use the Trap Receiver snmptrapd, which is for Linux available by default. By calling “snmptrapd -Lsd -Oq” the program writes received notifications with symbolic OID's in the system log file "/var/log/daemon.log", which is then used for log file analysis; or take the command “journalctl -f -u snmptrapd” or “journalctl -u snmptrapd”.

Note: For Windows, there is the setting up of "Event to Trap Translator" (evntwin.exe). It sends entries in the various Windows event logs as an SNMP trap version 1. The event id of Windows is represented as a Specific Enterprise Trap ID.

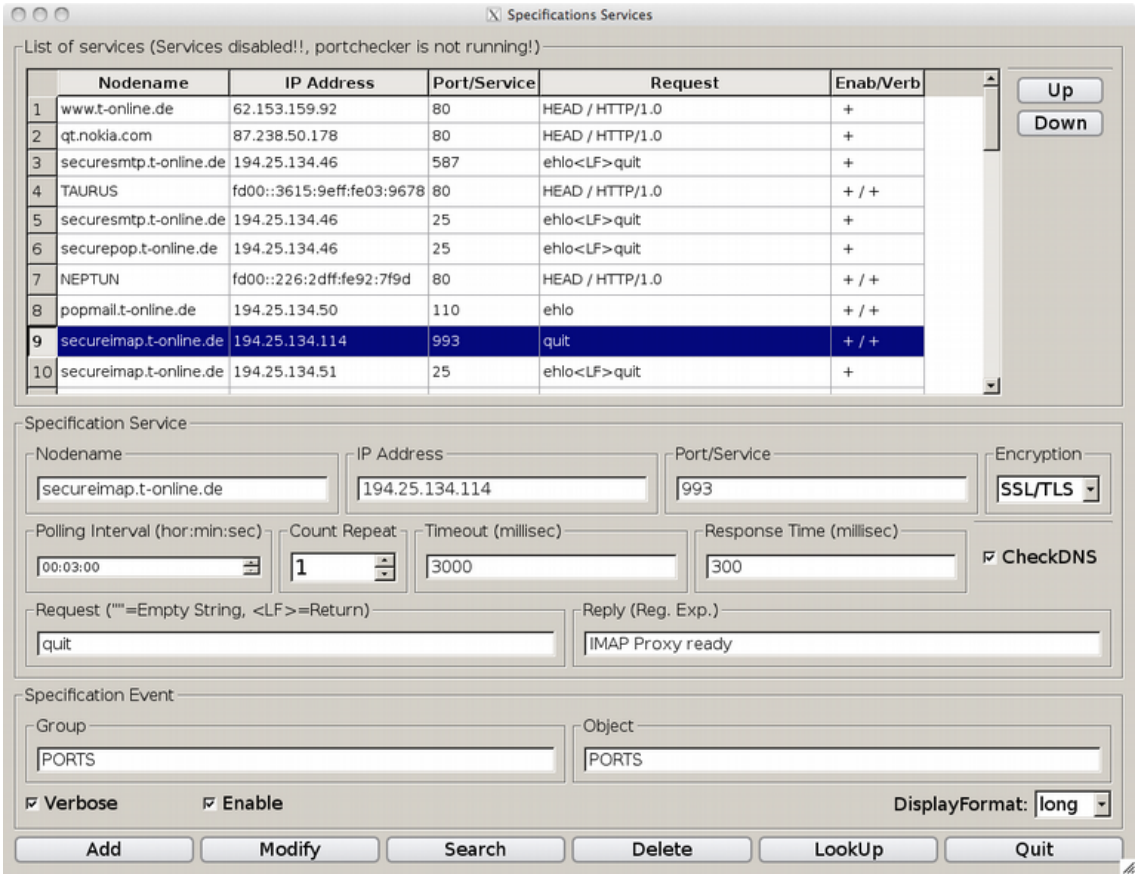
For SNMP requests (port 161), see chapter "Monitoring Scripts".

6. Active Monitoring

Active monitoring means the checking of ports (tcp) of remote servers. The test is carried out by the Management Station. There are two different options:

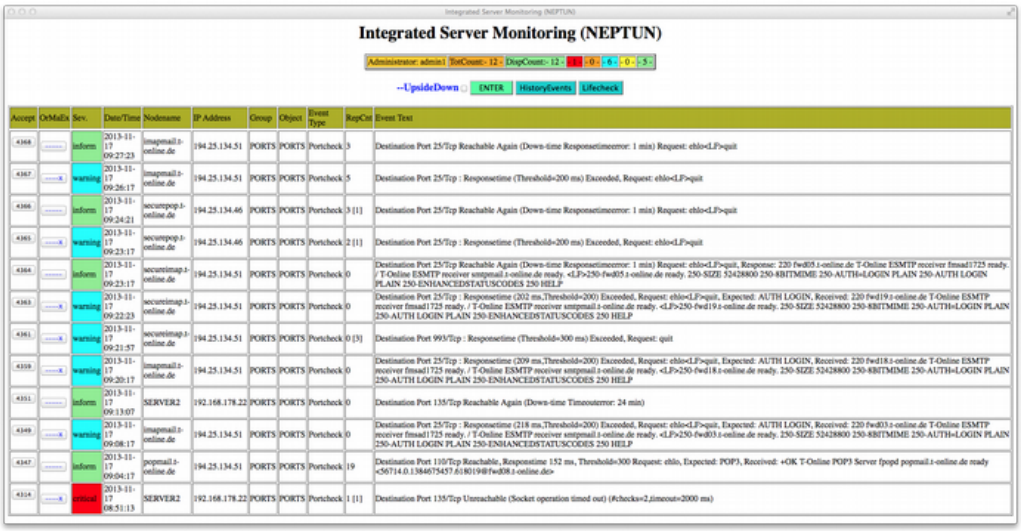
1. Check accessibility of the port by opening the network connection
2. As in item 1, in addition send a request and evaluate the return

In both cases, the response time is determined and compared with a threshold value. Moreover, it can be determined whether and how often the test should be repeated. It can also appeal to ports or services that use SSL encryption (for example, port 443 for https).



The figure shows the settings, an administrator can make to the Management Station. The checkbox "CheckDNS" causes the node name entered to be used as a function argument. Thus, the DNS name resolution is also examined. Apart from that, the IP address is taken as a function argument.

The output and the format of the output are set by the system and can be adjust by the filtering mechanism.



The figure shows messages from the monitoring port in the web display. After a disturbance, the duration of the outage (down time) is displayed with a green message.

7. Mass problem and data management

One of the most important requirements is the central display and management. This means that there is just one (possible) Management Station and not multiple, even if the number of servers to be monitored is significant, regardless of the number of monitors per server. This must also apply when the network environment is difficult due to firewalls, (double) address translation and other features.

For the present system, the maximum number of nodes per Management Station is **8192**. If several stations are in use, they can exchange messages with each other.

On account of the resulting strict requirements for the run-time efficiency a specially developed solution to this problem consisting of a combination of shared memory and indexed sequential binaries is used rather than a (relational) database management system.

When dealing with the management of messages and events a distinction has to be drawn between the current events, not yet processed, and the old messages, the history data. The current messages are held in a shared memory area, which is organized as a **ring memory** and has a capacity of 100,000 events. Here, all comparison and substitution operations are performed when new messages have arrived. In parallel, there is the archive file for the long-term storage of old messages. It is a binary file, whose capacity is – in case of a 64-bit system - practically only limited by the size of the file system. Thus, it is possible to trace back old messages for years, even in larger environments.

It is important that both the part of the shared memory and the binary are directly available for access to the X11 interface and to the Web interface without any transformation. The access times are accelerated by binary search, so that a large number (several million) of archived messages can be handled efficiently.

Data reception from the nodes is organized so that the acceptance and then the following processing are decoupled, with an internal buffer as interface. Both processes are carried out concurrently in the form of threads. Due to the asynchronous processing and storage load peaks, as they may occur during operation of several thousand clients, are offset.

Data management is fully integrated into the system, there are no adminis-

trative expenses as would be required in a regular database management system. The data areas are expanded dynamically during operation. Old messages can be retrieved and downloaded from the X11 and the web interface.

8. Agents for monitoring log files

Log files ("Logs") are text files that are continuously described by system and application programs with information about their condition. Logs are ubiquitous, at least on Unix systems. There is not only the system log file(s) (syslog) but also log files for database management systems, Web servers, firewalls, backup servers, etc. Each application, for which a server is operated ultimately, usually has one (or multiple) log files.

This system offers the possibility to evaluate a variety of log files on a server by filtering out such entries that suggest an acute or impending failure using appropriate search patterns. The search patterns use *extended regular expressions* (POSIX standard). Found records or rows are sent along with the file name to the Management Station, where they are displayed as a message text. If necessary, the message text on the agent and/or the Management Station can be changed by a format string.

Large files: Optionally, the size of a log file can be monitored. The maximum size in megabytes (MB) and a severity are given as a threshold. If the values are exceeded, a message with a boilerplate message text appears.

Exception handling: If the target file does not exist or cannot be opened read, a message with a boilerplate message text is displayed. This behavior can be turned off by using a special character in front of the declared file name.

The agents of this system use a list of filters that are set in the associated configuration files along with the file name. A filter consists of pattern, optional format string, severity, and optional counters for the frequency of entries found.

There are three different filters:

1. Positive filter: line corresponding to the pattern is displayed
2. Suppression filter: line corresponding to the search pattern, is suppressed
3. Negative filter: It appears when the entire search process encounters no line (search for absence)

Using the list of filters on agents, which may be arbitrarily long, a selection (pattern) and an assessment (severity) is performed. The process takes place at the point where the data arise. Data that are not selected make no

appearance, meaning no burden for either the network or the Management Station.

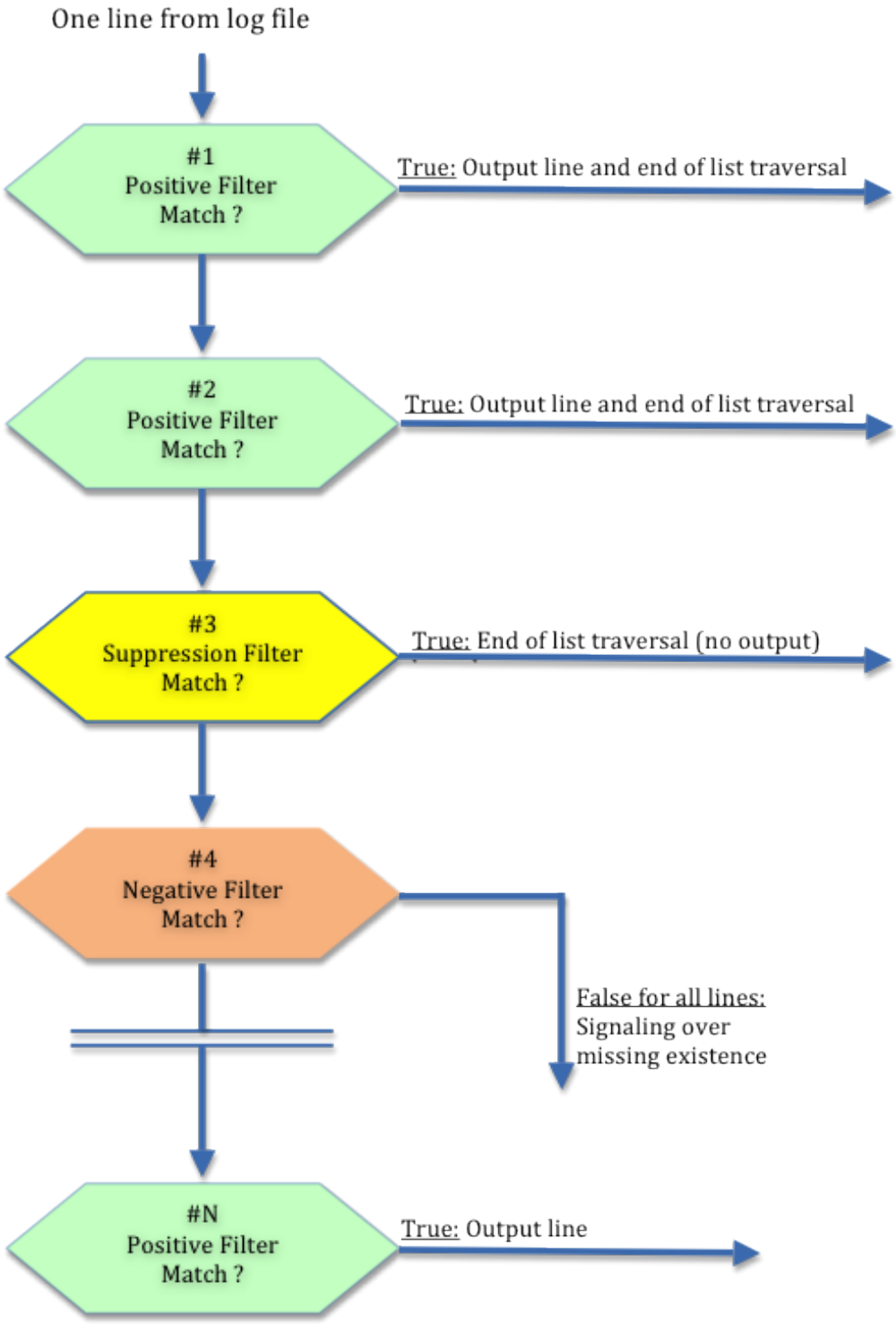
The list of filters is performed at each line or newly added row of the target file from the first to the last element. If a search pattern matches, the run ends. This form of processing ensures uniqueness, that is, it does not matter (it may even be intended) whether a search pattern from the list is contained in another. So for example the pattern "error" is included in the search pattern "noerror".

By placing "noerror" **before** "error" a suppression filter is provided, which prevents the appearance of unwanted messages. The combination of positive filters and suppression filters in an ordered list allows the use of general search patterns such as "fatal", "emergency", "panic", "sql-error", "segfault", "inconsistencies", "deadlock", etc.

When specifying the file names Meta characters such as '*' and '?' ("Wildcards") are allowed in their base name (Example: /var/log/*.log). These log files can be bundled in a common directory. In addition, newly added files, that match the pattern, are dynamically detected.

In the following diagram, the evaluation principle is shown again with a list of length N. It should be noted that the order of positive filtration and suppression filters is significant. For negative filters, the order does not matter, because the absence of the entire process is crucial.

The same mechanism is also used in the analysis of monitoring scripts.



Evaluation cascade for log files

The other attributes of a filter in addition to the severity are optional information about maximum and minimum occurrences. Thus, "message flooding" is prevented. In addition, it is possible to make a transformation of the search result in both the agent and the Management Station by using the format string.

The polling interval for the agent usually ranges from 1 to 5 minutes. In special cases, it is possible to reduce it to two seconds (program “asyncmonagent”, see below). Analysis of more than one line is possible. To this effect, the special character ‘\n' has to be used in the search pattern.

Examples of log files are the Unix system logs:
(“/var/log/messages“, “/var/log/system.log“, “/var/log/firewall.log“, usw).
The Apache Web server has the log files "error_log" and "access_log".

8.1 General log file analysis

For the various Unix derivatives there is the program logmonagent. For Windows there is the agent program logmonagent.exe.

The agents are parameterized by different configuration files. In a configuration file multiple log files can be set along with the list of filters.

Incremental analysis:

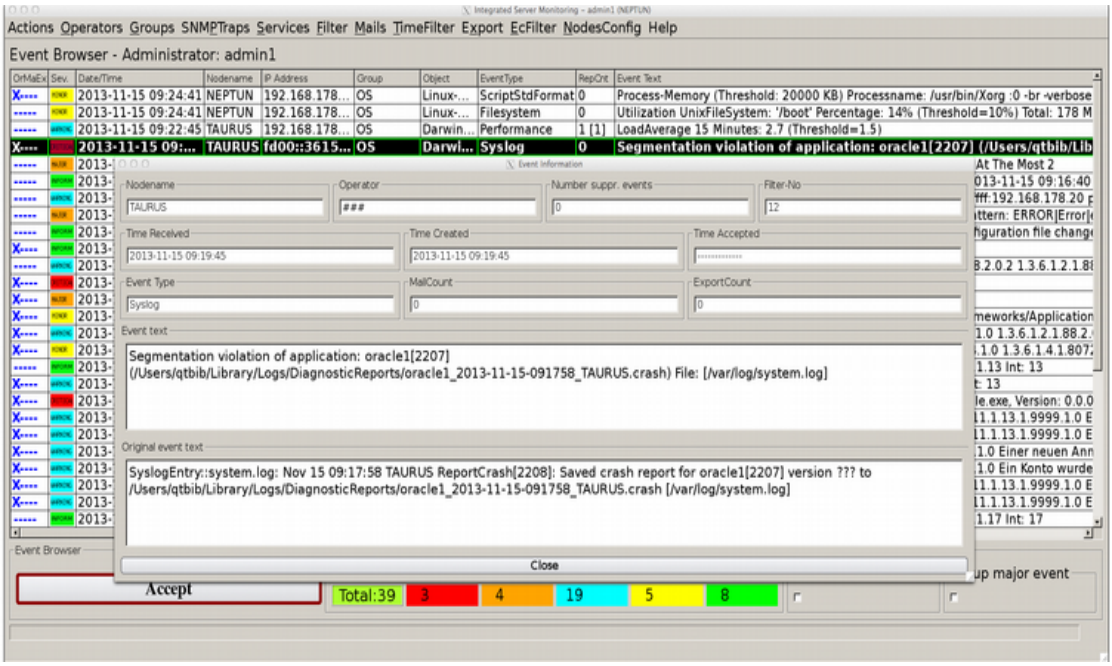
During the polling interval the growth of the target file is successively evaluated. This is the standard in log analysis.

Total analysis:

The log file is always viewed from beginning to end. The analysis is performed, if the modification time of the file has changed.

Analysis of this type makes it possible to check e.g. system files or configuration files such as "/etc /hosts", "/etc/exports", "/etc/sshd_config", etc. for suspicious or unwanted entries. At the same time you would know, if the files had been changed. Moreover, the deliberate absence of certain system files can be (for example, "/etc/hosts.equiv") monitored, i.e. a signal would indicate that someone is creating the file.

The same applies to Windows.



The image shows a message from the monitoring of the system log file "/var/log/system.log" of Mac OS X. The search pattern (search argument), which caused the message, is "[Ss]aved crash report". The line found, the search result, has been converted into a short, readable form by putting it through a filter with a format string at the Management Station. If you double-click on the message you will get the widget "Event Information" with the original message text.

This is the incremental log file analysis. See also the example in the Appendix.

8.2 Multiple log file analysis (Unix)

Another agent makes it possible to monitor not only one but several log files that are located in a directory. The list of filters applies to all files found. In contrast to the previous agent we define here the files that are to be evaluated, using a list of search patterns ("wildcards") for file names.

The name of the agent is: logdiragent and is parameterized by different configuration files.

Sample configuration file for multiple log file analysis: SAP/R3-Tracefiles:

```
destination::192.168.178.21 # Address of management station
portno::55555 # Port tcp for transmission
attribute::SAPGROUP SAP/R3 # Attribute Group, Object
pollingsecs::30  # run as daemon
logfiledir::/usr/sap/PD3/PWEBMGS11/work  # directory for log files
files::4[maj]::dev_w? dev_w?? dev_rfc? dev_rfc?? dev_disp dev_?? stderr*
# Specification of file names in directory, „4[maj]“: large files for files > 4 MB
# below the list of filters for the analysis
 filter::"update deactivated;;vb error: $*"-crit # after “;;” format string
"vb error;;vb abort: $*"-crit # vb error
"^Disconnecterror:"-crit "^Sqlerror:"-warn
"^Sevdberror:;;Error data base: $*"-maj
"^Profileerror:"-warn "Sharedmemoryerror"-maj "^Stackerror:"-warn
"^Mallocerror:"-maj "^Applicationerror:"-maj "^Inputbuffererror"-maj
"^Speichermangel:"-warn "^Shared Memory"-warn
"update activated"-null # Suppression filter
"Error Code"-min # Job aborts
"Error in ABAP statement"-maj
"[Mm]emory exhausted:"-maj
"Error.*in.*application.*program"-min
"CPIC-Error"-null # Suppression filter
“ERROR.*shmctl“-maj “ERROR.*shmget”-maj
“ERROR|Error|error”-warn[-1] “WARNING|Warning|warning”-min[-1]
“FATAL|Fatal|fatal”-maj[3] # drift net with unknown lines “FATAL...”
# end of configuration file
```

The search pattern (regular expression) begins with a quotation mark '"' and ends either with ";;"or, again, with a quotation mark.

8.3 Multiple recursive log file analysis (Unix)

Log files are not only monitored in (just) one directory but also in the underlying directories.

The name of the agent is: logrecagent and is parameterized with different configuration files.

9. Agents for Standard Monitoring

The standard monitoring offers a number of ready, problem-oriented features that have been selected so that they would be relevant and sufficient for the vast majority of nodes. At this point, neither a programming nor an exaggerated configuration effort is necessary. The agent creates the design of the message texts, which can be adjusted using the filter on the Management Station, if required. The boilerplate message text contains not only the set threshold, but also the determined current values so that trends can easily be identified. The thresholds themselves are, as far as possible, related (percentage) values in order to achieve comparability and generality. See examples in the appendix.

9.1 Unix

The following functions are enclosed in the agent for standard monitoring:

- **Threshold file systems:** a general threshold for the utilization rate (ratio of occupied to available memory) is set that applies to **all** current and newly added file systems. In addition, it is possible to determine individual thresholds for each file system or else to remove it completely from the monitoring. In case an individual threshold has been defined, the existence of the mount points will be checked simultaneously. If a threshold is exceeded, each file system has a separate message containing the following information in the message text: mount point, threshold, total memory, free memory yet, file system type, and rate of change of consumption in MB/h. Thus, it is possible to estimate the time remaining until full occupancy of the disk.
- Threshold inodes of file systems: The percentage utilization as the ratio of the number of used inodes and total number of inodes is monitored for each file system. The threshold value is a constant 95%.
- **Process Monitoring:** Check if background processes are active. The test relates to the existence of a Boolean value or the number of instances. A list of process names is to be entered; search patterns (regular expression) are allowed. The search also refers to the process parameters. After failure and subsequent restart of a background process, a message appears automatically that explicitly indicating the failure time (down time).
- **Restart function:** extension of the process monitoring; in case one (or more) background process(es) fail(s) the process automatically restarted by a restart procedure which is indicated at the Management Station.

- Threshold number of zombie processes: Two thresholds + Severity can be set (e.g. > = "warning" 100 and> = 200 "crit").
- **Monitoring of syslog files:** Incremental analysis of one or more log files. Besides the official syslog file there may be others such as "secure.log", "daemon.log", "kernel.log". Furthermore, other log files, such as web servers, database management systems or applications are permitted.
- Threshold syslog file size: If the size of the syslog file exceeds a certain value in MB, a message appears. Value + Severity can be set. General monitoring of file sizes ("large files") is done in the log file analysis.
- **Monitoring programs:** call and evaluation of one or more monitoring scripts or executable programs. The analysis refers to the text output of the program, which is filtered exactly in the same way as the contents of a log file (see also monitoring scripts).
- **Listenports:** check local tcp ports to the loopback interface. Input a list of port numbers (e.g. 22, 80, 443). In this test, the ambiguity is avoided (network or server) that inevitably occurs during a test over the net. After loss and re accessibility, there is an informational message indicating the failure time.
- CPU Utilization Threshold: Adjustable two percentage thresholds + Severity for short-term load and average load over a configurable period also, e.g. 120 minutes.
- Threshold Swap: ratio of occupied to a maximum of disposable swap space in percent. Set two percentage threshold + severity.
- **Threshold load average 15 min:** Two thresholds + Severity (e.g. > 10 "warn", > 50 "crit") can be set.
- Reboot and Startup: message to the Management Station at this event.
- **Lifecheck (heartbeat):** Dynamic registration of the nodes on the Management Station. After failure of the server, such as maintenance work or a major disruption, downtime is explicitly specified on the Management Station.

It is also worth mentioning that - with regard to the important file system monitoring - it is not necessary to enter each file system name individually, which would be a significant problem if a few dozen (or more) names were to be entered.

The name of the agent program is for all Unix derivatives: basemonagent. It is parameterized with different configuration files.

9.2 Windows

The following functions are included in the standard agent for monitoring:

- **Threshold File Systems (Drives):** As with the Unix file systems. Enter a general threshold from which reports are produced. The local drives are viewed. In addition, each drive will be provided with an individual threshold + Severity. In this case, there is a critical signal for the non-existence of the drives. The second threshold of 98% causes a critical message. Output is similar to that of the Unix file systems.
- **Monitoring tasks:** checking the existence of tasks. Input a list of task names. The number of instances of a task can also be checked.
- **Service monitoring:** detecting the existence of active and registered services. Input a list of service names.
- **System Event Log:** Log file analysis for disorders of the operating system. Search argument is the event ID and/or the Microsoft event type "error" and/or "warning". Each event ID can be provided with an individual severity. One or more Event IDs may be suppressed. The message strings are also included in the message text.
- **Application Event Log:** Log file analysis for events in applications.
- **Security Event Log:** Log file analysis for events in system security.
- **Monitoring programs:** call and evaluation of one ore more monitoring scripts or executable programs. The analysis refers to the text output of the program, which is filtered exactly in the same way as the contents of a log file (see also monitoring scripts).
- **Listenports:** check local tcp ports to the loopback interface. Input a list of port numbers (e.g. 22, 80, 135, 443). After loss and re accessibility, there is an informational message indicating the failure time.
- CPU Utilization Threshold: Two thresholds for a short-term loading and for an average load can be set and are adjustable over a period of time (e.g. 60 minutes). It monitors all CPUs individually.
- Memory Threshold: Is defined as the ratio of occupied to total space. There are two adjustable thresholds + Severity (e.g. 90% + "warning" 99% + "crit"). When the threshold is exceeded, the total space in MB is also displayed.
- Threshold Swap (Page): Is defined as the ratio of occupied to the maximum size of the page file. There are two adjustable thresholds + Severity (e.g. 95% + "maj" 99% + "crit"). When the threshold is exceeded, the maximum size of the page file is also output to the Management Station.
- Reboot and startup of the operating system are signaled on the Management Station.

- **Lifecheck:** as Unix

The name of the agent program is: winmonagent.exe. See also the example in the Appendix.

10. Agents for Monitoring Scripts

Special agents for the implementation and subsequent evaluation of monitoring scripts to realize special requirements. The programs may contain instructions for error handling.

10.1 Unix

The programs scriptmonagent and asyncmonagent may be used to execute any type of monitoring scripts which can be parameterized with different configuration files.

Any executable programs or scripts that have a text output to stdout and/or stderr can be used. The output is filtered according to the same pattern as in the log file analysis. For every filtered line of text, an event is generated and sent to the Management Station.

In a configuration file, several programs plus filter can be arranged. The scripts can be provided with input parameters. If the exit code of a script is nonzero, the agent interprets this as incorrect behavior and will send a separate, critical message to the Management Station. In the event of a blockage there is a timeout (default: 30 seconds) and also an exception message. Similarly, there is an error message if the specified script is not executable.

The following serves as an example for a configuration file showing how the memory consumption of processes with two thresholds is monitored.

```
destination::192.168.178.21 # Address of Management Station
portno::55555 # transmission port
attribute::Linux Scripts # Group Object of the events
pollingsecs::600 # run as daemon
cmd::ps -efly | awk 'BEGIN {lim1=100000;lim2=500000} {if((NR > 1) && (NF >= 14)){
if($8 >= lim2){print "Lim2:",lim2,$0
} else if( $8 >= lim1 ){
print "Lim1:",lim1,$0
}}}'::"^Lim1:;;Process-Memory (Threshold: $2 KB) Usage: $10 KB, Processname: %15"-min[10]
"^Lim2:;;Process-Memory (Threshold: $2 KB) Usage: $10 KB, Processname: %15"-maj[10]
"*;;System error: $*"-crit[1]
#cmd:: ... more commands to evaluate
#end of configuration file
```

After the keyword cmd:: the command to run is entered ("ps -efly", Linux) edited via a pipe '|' using the Report Generator "awk", followed by the list of filters according to which the output per line is formatted and according to which two different severities are assigned depending on the exceeded threshold. For each process, which has exceeded the first or second threshold value, there is a special message specifying in its event text threshold, current value and process name. Of course, it is also possible to implement the command and (numerical) evaluation of a program, which is then declared in the configuration file. (The same function can also be implemented using the agent for standard monitoring).

Other examples for the use of monitoring scripts are the determination of the utilization rate of extents and tablespaces in the Oracle database management system. For Informix, it is the DB-spaces and logical logs.

The agent program asyncmonagent is an extension of the programs scriptmonagent **and** logmonagent. Each script or each log file that is declared in the configuration file can be provided with its own polling interval. The execution of the individual monitoring functions takes place concurrently in the form of threads. The agent is operated exclusively as a background process.

CRONJOBS (SCHEDULER) AND DAEMONS:

Using the agent program asyncmonagent also cronjobs can be realized. The handling of the times happens as when configuring crontab() on Unix.

Example:

```
destination::168.178.20.21
portno::55555
attribute::Linux AsyncMonitorCollection

#logfile::[name::]<crontabspec::<fileenam>::filterlist
#total::[name::]<crontabspec>::<filename>::filterlist
#cmd::[name::]<crontabspec>::<command>::filterlist
cmd::0 10 * * 1-5::echo “It is ten o’clock in the morning (mo-fr) `date`“::“*“-info
# check disk space
cmd::15,45 6-20 * * *::df -k::“[ ]100%[ ];;FS full: $*“-crit “[ ]9[5-9]%[ ]“-maj
“[ ]9[0-4]%[ ]“-warn “[ ]8[7-9]%[ ]“-min “[ ]8[2-6]%[ ]“-info
cmd::5::journalctl -f -q --since=-2m -p crit::“*“-crit # fastest way of signaling
# “journalctl -f“ will start after 5 seconds and does not terminate
# ... more functions and instances
```

SNMP:

The script agents can perform SNMP queries on the local server (domain name: localhost). This one takes the Unix commands snmpget, snmpwalk, snmpdelta, snmpbulk, and filters the output. The results reach the Event Browser passing the normal tcp port. This spares you continuous queries over the network.

Direct messages:

In addition, there is the command rsendmsg that can be used to directly send messages to the Management Station in scripts of any kind (Shell, Perl, etc).

10.2 Windows

There are the programs scriptmonagent.exe and asyncmonagent.exe, which have the same functions as for Unix.

Example: Evaluation of the command "netstat -an"

```
destination::192.168.178.21
portno::55555
attribute::Windows
pollingsecs::120
cmd::netstat -an::“^[ ]*UDP/v“-null “127\.0\.0\.1|\[\::1\]“-null
“:(68|13[78]|500|1900|3702|4500|4919[23]|515[45][1-9]|5855[01])[ ]/v;;Udp-Port: %1“-warn[3]
#cmd::... more commands
```

The analysis is done with a positive list of allowed ports, which is part of the third filter. The first filter suppresses all lines that do not begin with "UDP".

Direct messages:

As for Unix the command rsendmsg.exe is available for Windows. It is parameterized with the port number, the address of the Management Station and the components of the message.

11. Agent for Security (Unix)

Agent for the purpose of security that reports changes to the file system in short polling intervals.

The agent reports after the initialization change of the attributes of system directories and system files. These are preferably those files and directories that determine the behavior of the operating system and the change in operation only takes place under special circumstances. These include regular installations or updates, but also covert installations by ***rootkits*** and other attempts to tamper with. If desired installations take place, one can make comparisons between announced and real changes. When added new files of any kind in a directory, it is also reported!

The investigated distinguishable attributes are: *inode number*, *size*, *modification time*, *mode/permission*, *status time*, *user id of owner*, *group id of owner.*

The name of the agent program is secmonagent and is parameterized with different configuration files.

Example:

```
destination::192.168.178.20
portno::55555
attribute::Debian Security
pollingsecs::30 # run as daemon, evaluation every 30 seconds

files::/bin[crit] /sbin[crit] /usr/bin[maj] /usr/sbin[maj] /etc/init.d[crit] /etc/[maj]
/usr/lib /boot[crit] /boot/grub[crit] /lib[maj] /lib/modules[maj] # ...
exclude::/etc/mtab /etc/resolv /etc/adjtime # ...
```

It is declared a list of directories and/or files that are optionally supplied with a severity. Optionally you can with the keyword exclude:: declare a list of file names and/or subdirectories with full path name to be excluded from monitoring.

The agent messages include in the Event Text the names of the directories and files located in them as well as the kind of change.

12. Lifecheck (Heartbeat), Dynamic Registration

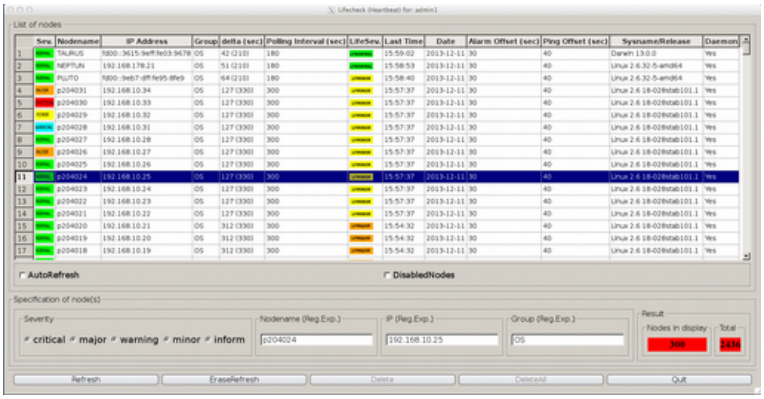
The operation of the monitoring agents does not require the input of any data such as computer name and IP address. These are automatically recorded and managed on the Management Station. Information comes from the servers to be monitored and is constantly updated.

A node is detected by the system after the standard monitoring for the server has been set. With each call a special invisible control message - which also serves as an authentication - is sent to the Management Station renewing a timestamp. The age of the timestamp is monitored continuously in the background. Both the (first) application as well as the lack of control message is signaled after a certain configurable number of seconds in the Event Browser.

In addition, the Management Station automatically performs a test with Icmp-Ping, if a server has not reported after a certain time, which is also adjustable. After a total of three test with a negative result and the appropriate signaling output the server automatically goes to the state of "disabled". After that, there are no more messages. The server remains in this state until it logs on again. When a new registration has happened, an informational message is sent indicating the failure time in the message text.

This automatic feature allows control over the network connection and the potential failure of the entire server.

The number of servers monitored can be listed in the X11 and web interface, combined with a **Status Indication** that shows the highest severity message in a browser for each active node. If a node name appears with a different IP address than previously reported, an automatic warning is generated in the Event Browser. This may well happen for servers with more than one network card. The following screen shot shows a trouble-free running server that does not cause messages indicated by a periodically renewing timestamp and a green status message (left column of the next picture).



The picture shows the list of nodes monitored by the agents of this Management Station. For each server it can be specified when it has answered the last time, the interval in seconds of the current time and the polling interval. The input fields in the lower part are search boxes, as the list can be very long. With the checkbox "DisabledNodes" the servers in question are listed separately.

Note: In conventional systems of this type, the administrator must manually input host name and IP address on the Management Station. From then on, they are virtual system constants forming the prerequisite for monitoring. The problem is less the amount of work (plus option to the wrong input), but rather the fact that the host name and IP address have already been declared elsewhere, namely in the original. It is also not uncommon that a server has more than one address. Another point is the dependence of DNS records. This inflexible declaration becomes a serious problem in case of certain server architectures (for example, high-availability systems) that provide dynamic allocation of addresses and server names.

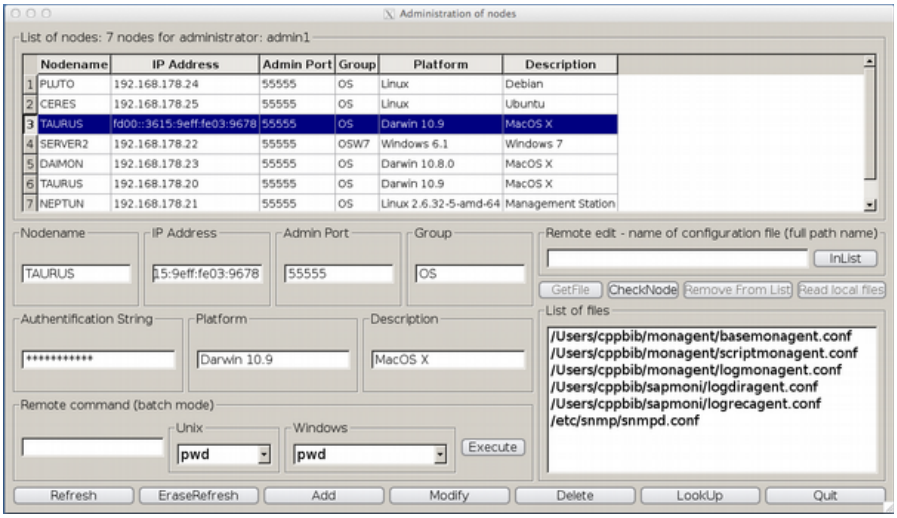
13. Configuring the agents, Command-Interface

The graphical user interface offers the possibility to centrally edit the configuration files and to manage the nodes. For this purpose, it communicates with a background process on the server via an udp/tcp port. The names of configuration files are kept on the Management Station whose content is then edited via the Web. There is no redundant data. After processing each file is restored. There is no functional dependency on the communication of the monitoring agents.

There is also the possibility to save configuration files locally on the Management Station and to read existing files, which can then be distributed to any node.

Moreover, there is the possibility to execute commands for the operators run on a node (for example, “ps -ef”, “df -k”, “netstat”, etc.) - for administrative purpose - via a command interface. The text of the command return appears in a separate widget. See also the example in the Appendix.

Authorization for the configuration lies with the administrator and with the operators the server is assigned to.



In the picture on the right below the names of the configuration files of a selected server are listed. A double-click on the upper left column of the list takes you directly to the server’s command interface to issue commands.

The name of the background process is available for Unix: remoteconfd, and Windows: remoteconfd.exe.

Security: The interaction is particularly protected

- Authentication by secret keys, IP address of the Management Station and/or "authentication string"; provides protection against replay attacks
- AES encryption with cipher block chaining mode (CBC) and *session key*
- Checksums for data integrity

14. User Management

You have to log on with a user ID and a password to use the system. The user management is the task of an administrator. There are three input screens on the graphical user interface.

1. Declaration of the user, in this case, the name and the initial password is entered and determines whether there is an operator or administrator
2. Declaration of group names
3. Assignment of groups to an operator

Up to 255 operators or administrators can be created. The number of groups is not limited.

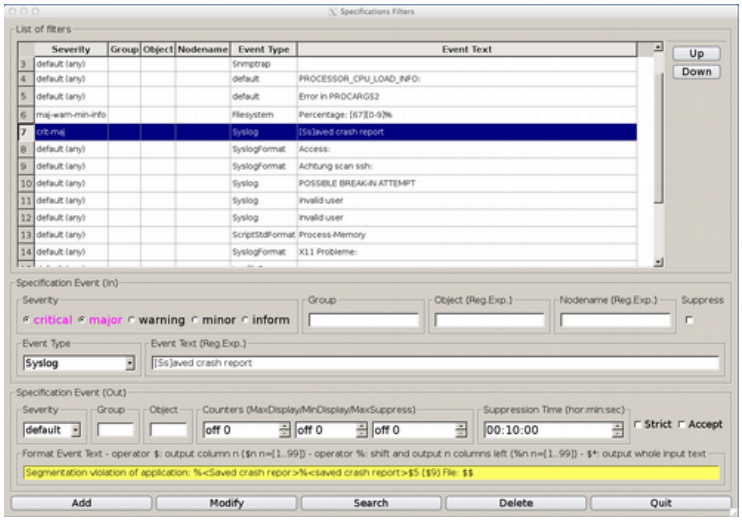
An administrator sees all messages, an operator only the messages of the groups assigned.

15. The filter mechanism (Management Station)

Integral part of this system is the filtering process of incoming messages at the Management Station, which is aimed at keeping the system clear of not relevant or repeating information and/or at automatically archiving messages no longer relevant by the system. The respective configuration is done by an administrator on the graphical user interface in operation.

15.1 Pre filter mechanism

The pre filter mechanism is located between the receiving and storing a message. Events that are suppressed do not get into the database. In his context, it often occurs that similar messages are coming from the same source and differ with regard to their attributes just in the message text. For example: In case of performance data the input text varies often only in numerical values. The degree of similarity can be determined through the use of search patterns (regular expressions). In addition, the event text can be changed by using a format statement.



The figure shows the graphical input mask for the filters. The upper part contains the already existing filters. In the area below ("Event Specification (in)") a message with its attributes is described. Free input fields (or "default") cause a positive comparison result for this attribute. With the checkbox "Suppress" messages can be suppressed indefinitely.

In the lower part of the mask ("Event Specification (out)"), you can optionally specify the outgoing message by defining or redefining "Event Text", "Severity", "Group", "Object". With the input field "Suppression Time (hor:min:sec)" you determine the duration of the temporal suppression of the same messages. In case of suppressing, a counter at the respective message is incremented by one, which is displayed in the column "RepCnt" of the Event Browser in square brackets.

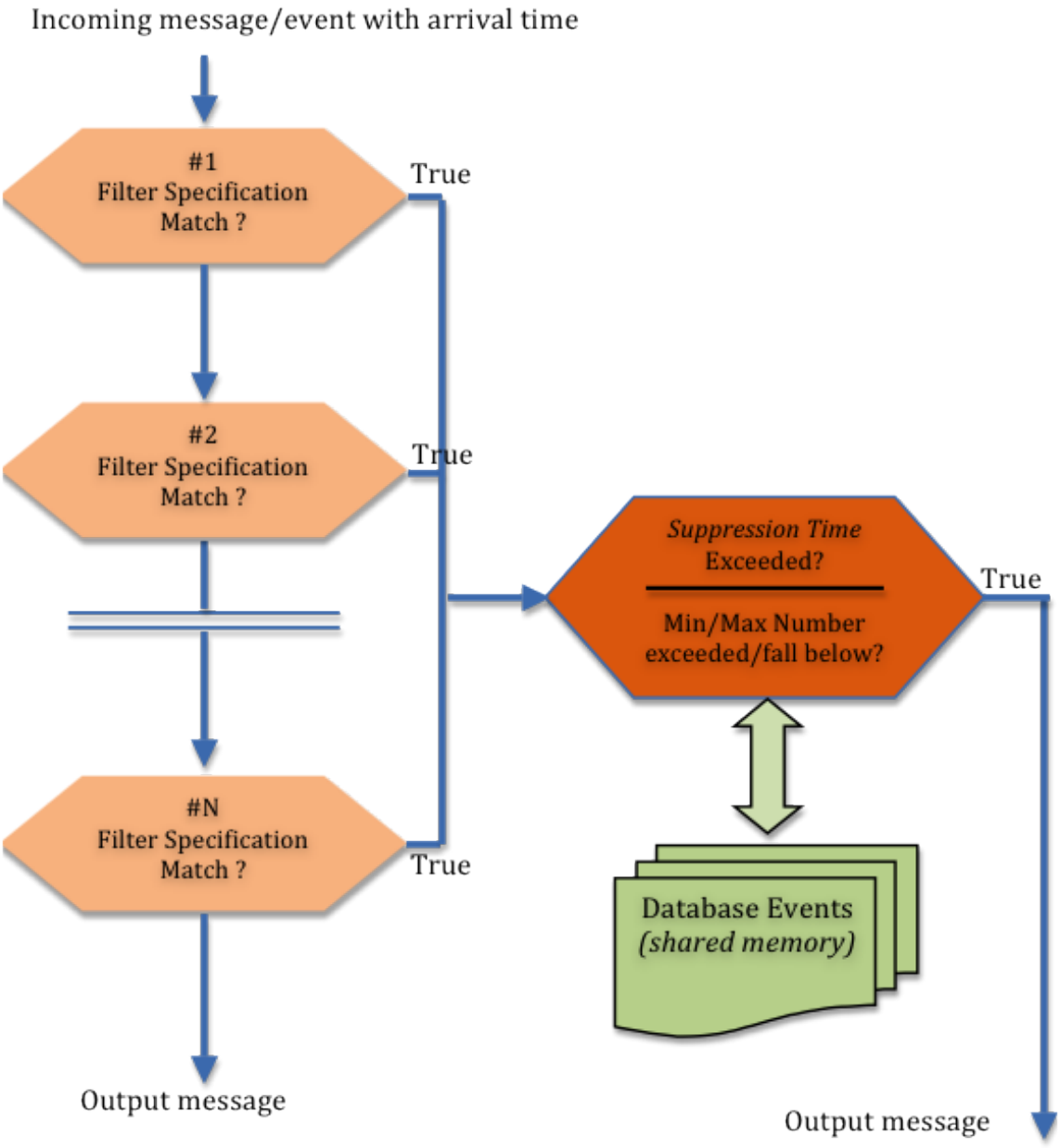
If all input fields are left blank or "default" and all severities are set, it has the following effect: Each message that hits this filter is suppressed for the set period, if repeated. This general description of an incoming message can gradually refined by placing appropriate filters in front. This is a self-regulatory mechanism that detects even previously unknown messages.

Important: This affects only messages with the same content, occurring repeatedly over a certain period of time while different messages are displayed (**Difference Display**).

The checkbox "*Strict*" turns off the comparison of the message text. Similar messages are also suppressed. Using the spin boxes in the fields "*Counters*" the suppression of the same or similar messages in number is possible.

When a message arrives, the filter list is traversed (starting from the top). If they coincide with the specification, there is a treatment, and the process terminates. It should be noted that the result of filtering depends on the fil-

ter order. They are to be arranged so that the special filters precede the general ones.



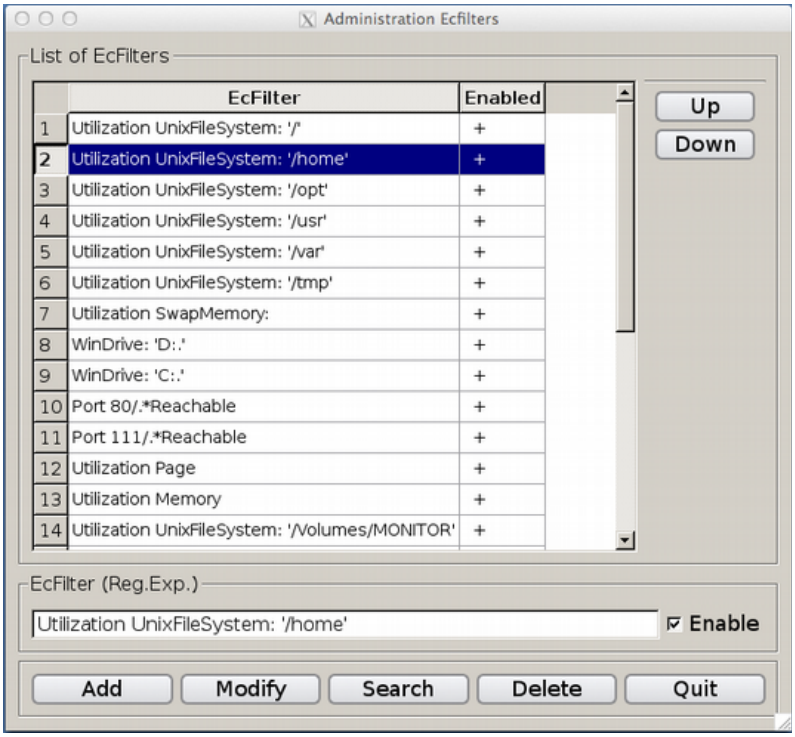
Again, the graph shows the principle of the processing in a number of N filters. It takes place in two steps: firstly to determine the filter by comparison and secondly to decide whether to display or not on the basis of already existing messages. In this way, the system is able to dynamically adapt the message volume, without causing a loss of information.

During all transformations the original message text is preserved and can be viewed by double-clicking on a message.

The number of filters is not limited. If the list of filters is empty, each message is displayed.

15.2 Filters Downstream (EcFilter)

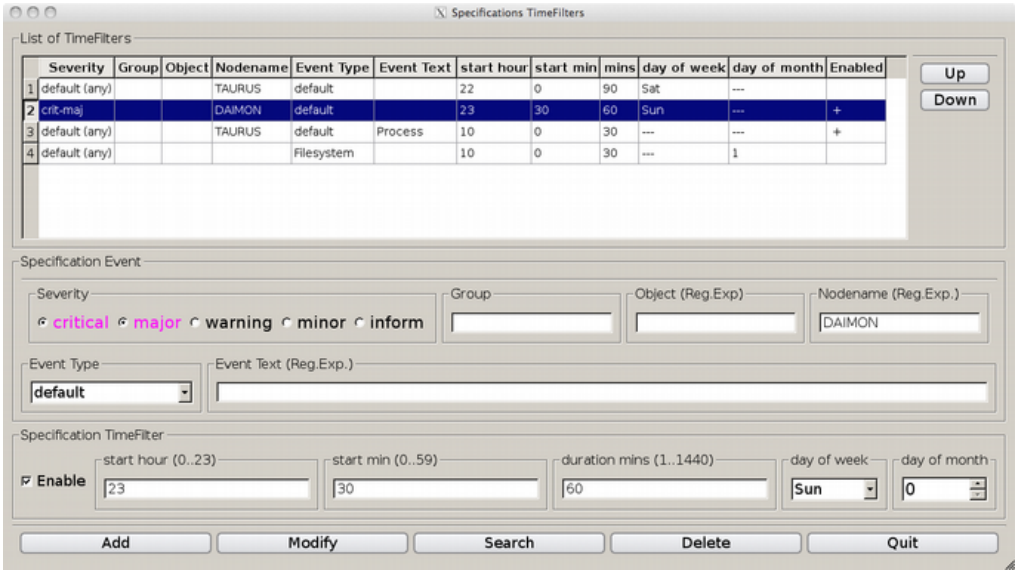
Here, it is possible to automatically archive similar messages that are still up to date and replacing them with the newly incoming message by using a list of search patterns. To make this process visible from the outside, the repetition counter ("RepCnt") of the corresponding message is incremented by one. Identical messages are always treated like that.



The figure shows the graphical input screen. Here, input a string constant or a search pattern (regular expression). Thus, the message text of a new message will be compared to existing events thus determining an *event correlation*.

15.3 Timefilter (Scheduler)

Here, messages are suppressed in a set time frame. This applies for example to maintenance periods scheduled for a particular day of the week or month in a given period. Another example is the operating hours on working days only between 6 a.m. - 8 p.m. However, it is principally assumed that a server is running 24/7.



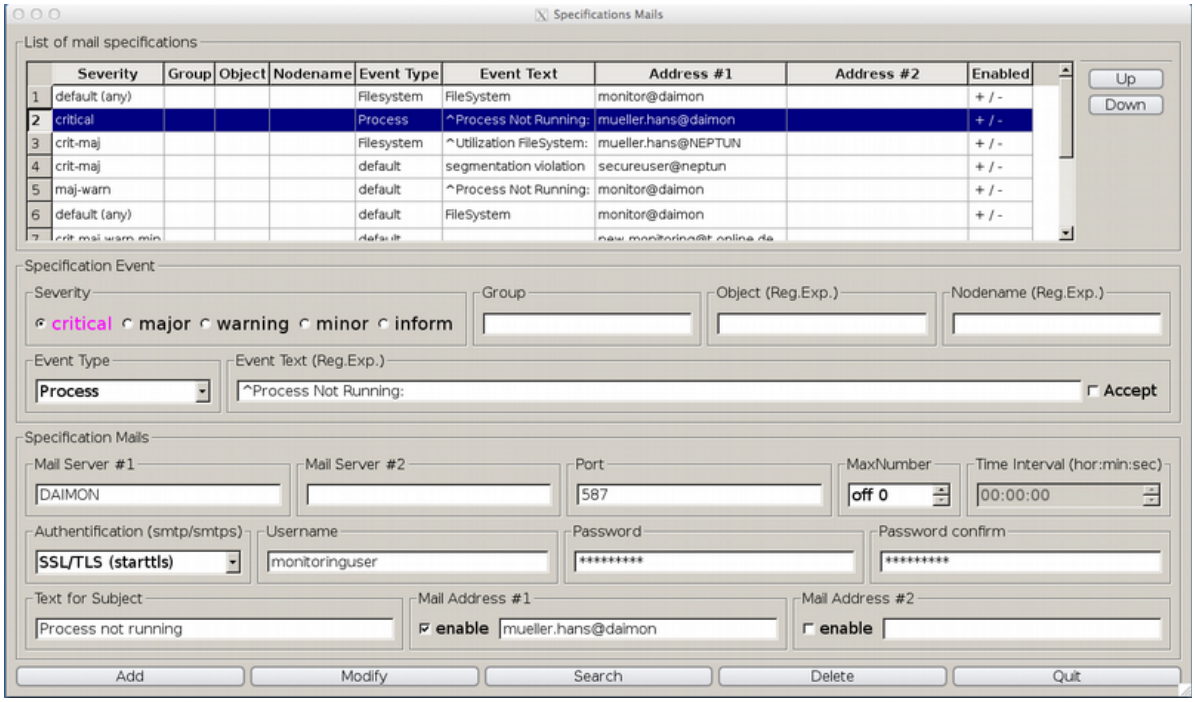
The picture shows the input mask of the time filter. In the upper part the existing filters are listed. In the middle section, messages in their attributes ("Severity", "Group", "Object", "Nodename", "EventType", "Event Text") can be specified. In the input area below, the period, in which one specifies a time per week or month and the duration in minutes, can be defined.

16. Forwarding of Messages

Accrued messages can be routed by a special and a general device in real time. It should be noted that the display and storage of messages and their escalation are two different processes (one can not replace the other).

16.1 Forwarding by E-Mails

Mails via SMTP or SMTPS (secure SMTP) can be directly generated without programming. Ann incoming message is specified adding mail address(s), mail server, backup server, and optional header text. In the case of secure SMTP user ID and password of the mailbox on the mail server have to be added, too. With the input fields "MaxNumber" and "Time Interval" the number of outgoing mails in a period can be limited. In the email, the message is displayed as a text component by component.

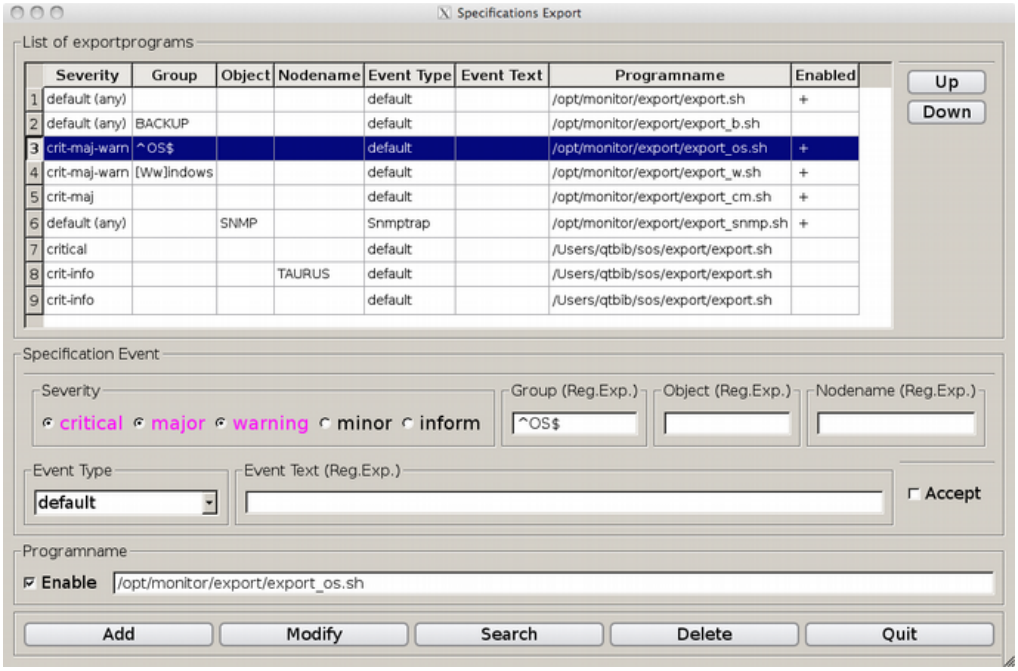


The figure shows the specifications for sending mails. A message can be sent to any number of addresses.

16.2 Export (Automatic Actions)

The system may export messages that are specified in the graphical user interface, a program interface. The processing program is also specified in the user interface. The message itself is passed, component-wise, as positional parameters to the program. This functionality is often referred to as "automatic action".

This mechanism can also transmit messages to another Management Station. The transfer can be done using either tcp or udp.



The picture shows the graphical user interface for exporting messages.

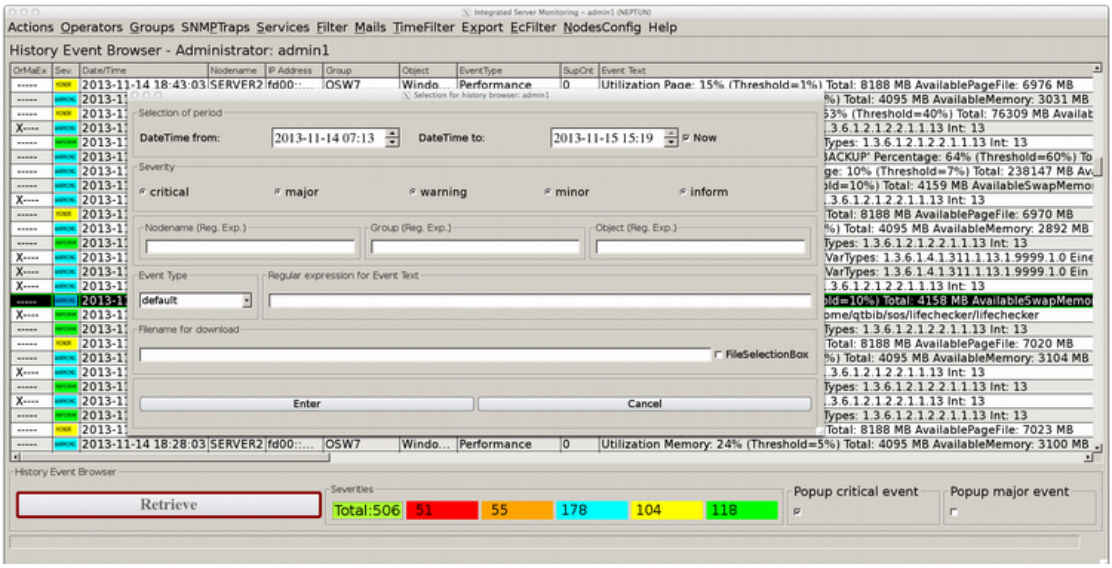
If the program you are defining a return code other than zero returns, this is regarded as an error. In the first column of the Event Browser (column heading: "OrMaEx") is to see if the message was sent as email and/or whether an export took place. If forwarding was successful, the letter 'X' appears, on failure comes the letter 'E' in red.

17. History Data (Reports)

After a message has appeared and been processed in the active X11 Browser or the Web Interface the operator or administrator presses the push button "Accept". In this process, the system records the time and the name of the operator. The message is then no longer visible in the output medium, also not for other users.

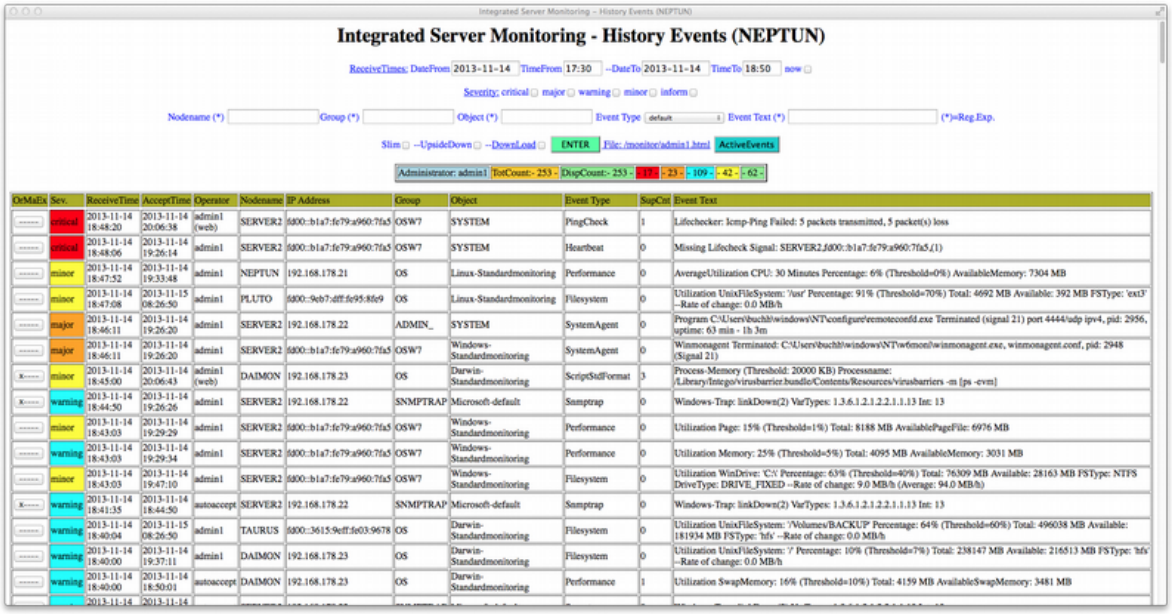
Thus, it can be controlled who has completed at what time which type of event (acknowledged).

Via a selection mask in which you can look for certain attributes of an event, the old messages are made visible again.



The figure shows the selection screen, and the representation of the found events in the History Event Browser. Additionally, you can specify the name of a file in which the messages are stored line by line. There is also a file selection box for selecting the file name.

By double-clicking a line you get a widget with the times for creation, reception and acknowledgment of the message. Using the push button "Retrieve" a message can be placed on the active browser.



Analogous to the X11 interface there is the web form for the display of the history data. The operator can download the content directly to a file on his PC. The file contains one event per line, in which the components of the message are displayed as text and separated by semicolons.

On the basis of the history data it is possible to make not only statistical evaluations but also compare new problems with old incidents to optimize the error handling. This is especially true for large data volumes if the events date back several years.

18. Background processes on Management Station

There are the following background processes:

- monlistener: Receives the messages from the agents, one instance per port number; also receives forwarded messages from other Management Stations; reception takes place concurrently and asynchronously
- lifechecker: Accomplishes the heartbeat and indicates if a server fails, processing is carried out concurrently
- snmptraplistener: Receives the SNMP traps, default for 162/udp4+6; reception takes place concurrently and asynchronously, also asynchronous name resolution of IP addresses; the background process allows an unlimited number of clients
- portchecker: Implements an active surveillance; processing takes place concurrently
- browserctl: Maintains data base and accomplishes emails and exports messages; processing takes place concurrently
- udplistener: Receives messages from other Management Stations over udp

Concurrency is achieved by multithreading. The coordination of data access is done with fcntl(), mutex, semaphores.

All Programs for reasons of efficiency in C++ (also applies to the agents).

19. Copyrights

The developer of this program is Wilhelm Buchholz. The source code is his property. The author has worked twenty years in the system management, ten years of it with different tools in this area.

There are agents for the following platforms: Linux, Windows, Mac OS X, AIX, HP-UX, Solaris. Other platforms can be included.

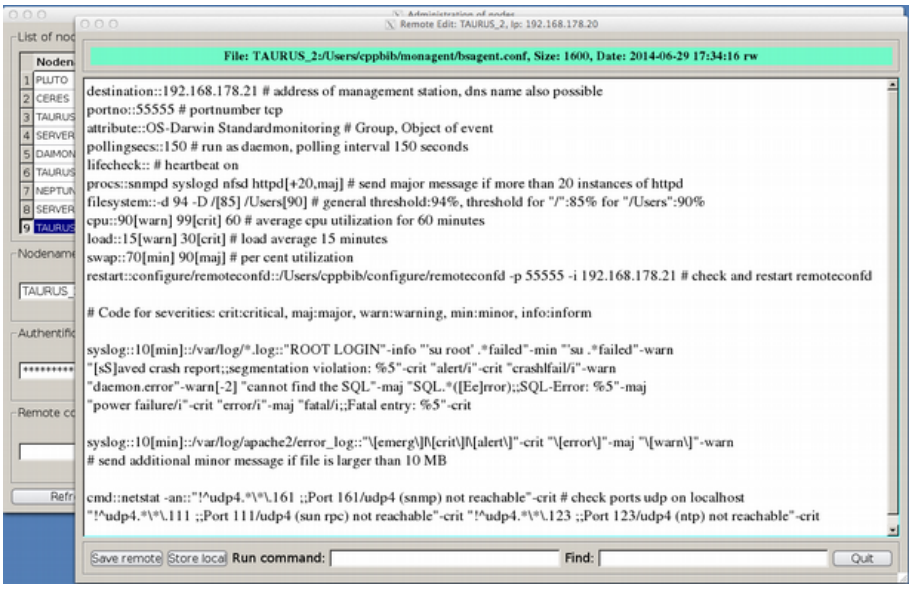
In the area of real-time monitoring Hewlett-Packard (HP) is available as a supplier, with the main product HP OpenView Operations (formerly OPC, then ITO, VPO, OVO) and IBM with the product Tivoli Monitoring in a variety of versions. Another supplier may be BMC with the product Patrol.

Other systems in this area, including open source, are rather uninteresting, because adequate functions for presentation of results, data management, multi-user capability, log file analysis, capacity problems are missing or not clarified. The problem of capacity is serious; so what happens when there are no longer a few hundred but several thousand servers under surveillance. Only vague statements can be made as to the benefits and operating costs.

The extremely complex systems from HP and IBM stick out of the mass of other systems mainly because they offer a full log file monitoring.

20. Figures (Examples)

20.1 Example Standard Monitoring Unix



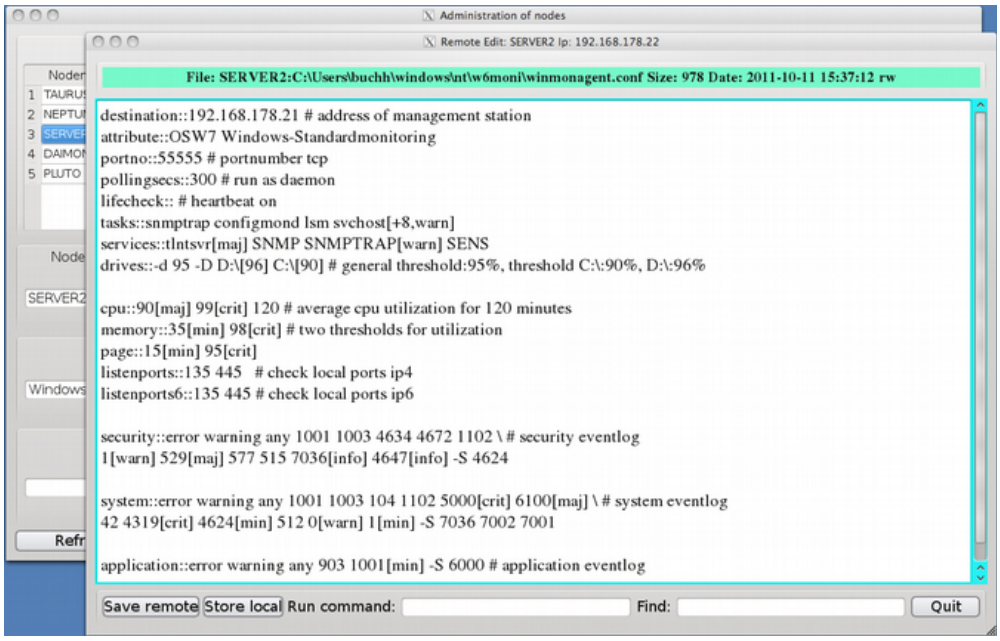
The figure shows a configuration file for standard monitoring in the remote editor of the Management Station. The agent is run as a background process with a polling interval of 150 seconds.

It is monitored: All file systems, a list of four background processes, swap/memory, load average, average CPU load for 60 minutes and all log files with the pattern "*.log" in the directory "/var/log " and the error_log of Apache.

Furthermore, the background process "remoteconfd" is monitored, and restarted should it not be active. With the command "netstat" the udp ports 111, 123 and 161 are monitored. The evaluation of the command is done with negative filters (lower picture).

The file system monitoring is one unit independent of the number of mounted file systems. The same also applies to the process monitoring and the port monitoring.

20.2 Exaple Standard Monitoring Windows

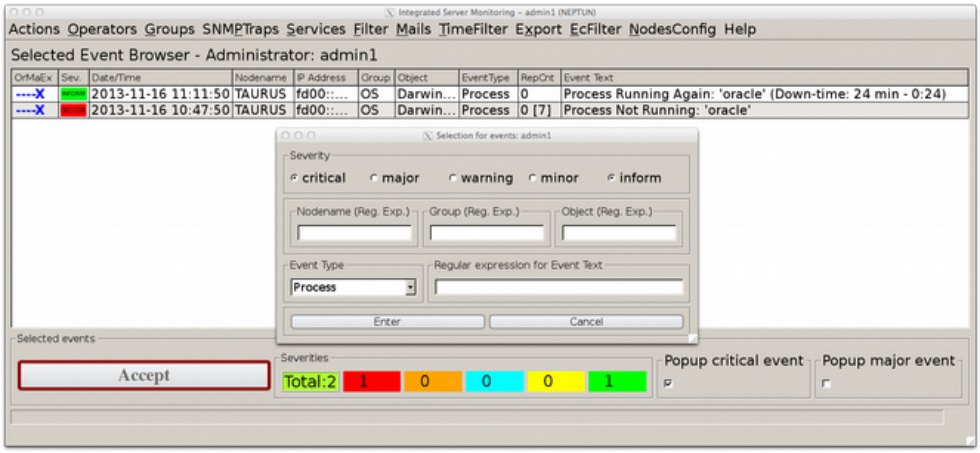


The figure shows the processing of a configuration file using the remote editor at the Management Station. This is the configuration file for standard monitoring for Windows.

Monitoring includes: All drives, a list of four tasks, five services, Memory, Page File, average CPU load for 120 minutes, two listenports ipv4 and two listenports ipv6. In addition, the System Event Log, the Security Event Log and Application Event Log. As a search argument for the event logs there are event IDs and/or event type. The option “-S” followed by a list of Event Ids causes the exclusion of the corresponding Windows events. The threshold values for the page file and memory are percentages of utilization.

It is quite clear that the configuration files are compact, easy to read and easy to manage, as they exist at the operating system level. By the exchange of configuration files good synergy effects can be achieved.

20.3 Example Process Monitoring

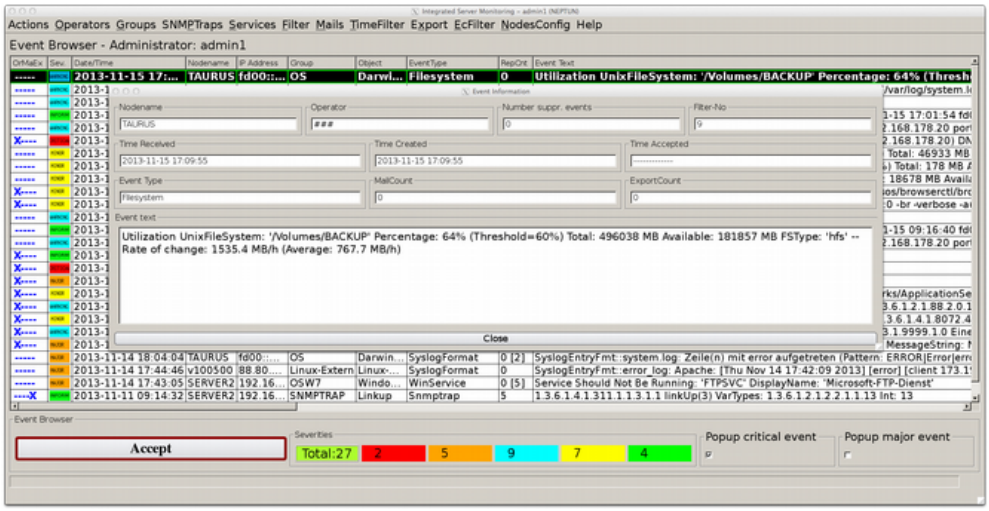


The figure shows messages in the browser "Selected Event Browser", which allows selecting specific messages in the active browser. The selected events indicate the beginning and the end of a fault.

At the beginning of the disturbance, the critical message comes with the text "Process Not Running: 'oracle'." The numerical value in square brackets in the column "RepCnt" indicates the number of messages, which have been suppressed by a filter due to substantive equality. If the background process runs again, a green (informative) message will appear automatically (without configuration) explicitly indicating the failure time in the message text. This information is usable for service level agreements and other statistical surveys.

The messages come from the agents and are standard monitoring (process monitoring). The same functions are related to tasks and services in the standard monitoring for Windows.

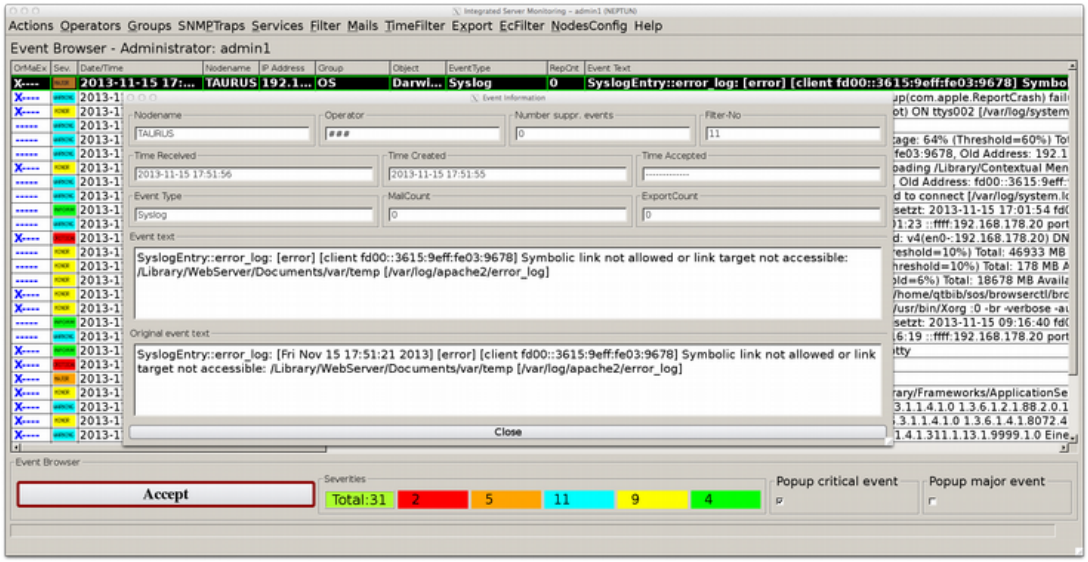
20.4 Example File System Monitoring



The figure shows the default message given by the file system monitoring. The message text contains the following information: Affected mount point, percentage utilization, exceeded percentage threshold, total occupancy in MB, available memory in MB, file system type, rate of change of consumption in MB/h, average speed in MB/h. By specifying the rate of change elaborate trend graphics are unnecessary.

All existing file systems are recognized without any additional effort. Each file system has its own message when exceeding a threshold value.

20.5 Example Log File Analysis

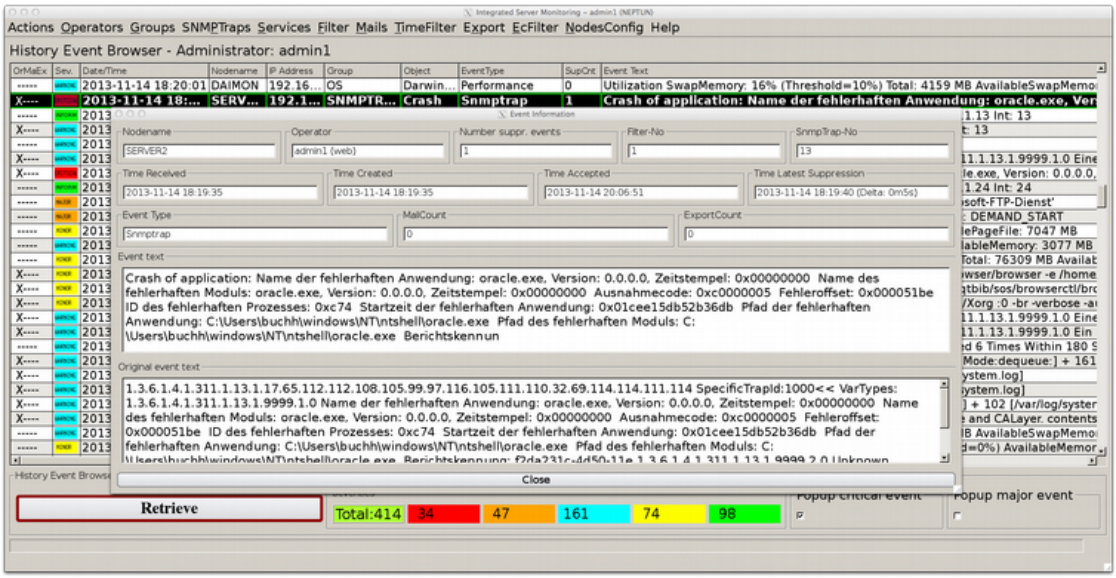


The figure shows a message from the standard monitoring, namely the analysis of error_log of Apache2. The triggering pattern at the agent was "\[error\]". The found line has been changed at the Management Station using a combination of substitution and left shift so that the dates have disappeared. You can see the original and converted message text in the image.

This substitution feature is of great importance, because, without having to bother with date and time, it is possible to concentrate on content (by value), equality and to suppress, temporarily, information. Thus, "message flooding" is prevented, which pose a significant problem for conventional systems.

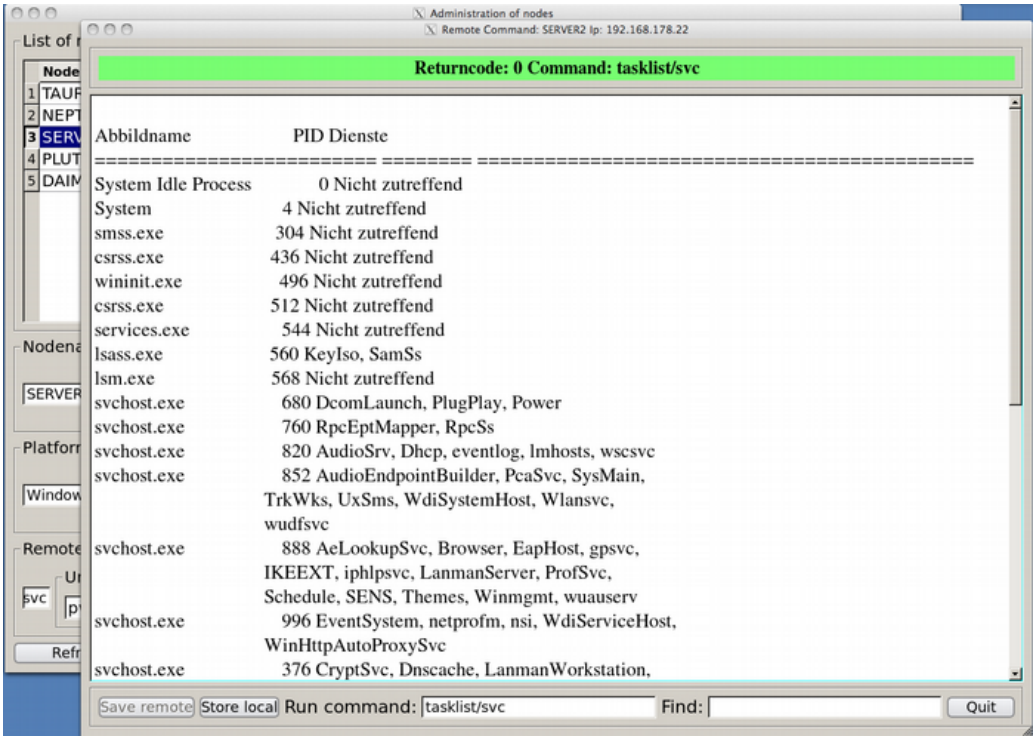
It may occur that in certain disorders (such as disk errors) thousands of entries are written to the system log file in a relatively short time. In principle, this may also occur with other log files. Therefore, an effective filtering in both the agents and the Management Station is necessary.

20.6 Example SNMP-Traps



The figure shows a trap message caused by a crash, which has been altered to a user-friendlier version by a format string. By double-clicking a selected message you get the widget "Event Information" showing the altered and the original event text.

20.7 Example Command Interface



The figure shows the return of a command to a remote Windows server. You can also restart services if the background process remoteconfd.exe has the appropriate rights. Communication is encrypted with AES/CBC. The same applies to Unix servers.