M/MONIT

User Manual

version 2.4

Tildeslash Ltd.

Contents:

Host	%Cpu	%Mem	Status
4-x64	5	3.5	No report from monit. Last report was W
h2	18.9	14	27 out of 28 services are available, 1 n
1	0.3	18.5	All 20 services are available
x86	0.5	2.2	All 9 services are available
x64	0.8	3.7	All 9 services are available
x64	0.5	4.4	All 9 services are available
-x86	0.3	3.9	All 9 services are available
x64	0.1	16.5	All 10 services are available
-x86	0.1	19.9	All 9 services are available
11-x64	99.3	45.4	All 11 services are available
s11-sparc	6.5	31.3	All 8 services are available
sd4-x86.localdomain	0.3	4.5	
ebsd70-x86.local			

Introducing M/Monit

M/Monit is a system for automatic management and monitoring of Information Technology Systems. M/Monit can monitor and manage distributed computer systems, conduct automatic maintenance and repair and execute meaningful causal actions in error situations.

M/Monit uses Monit as an agent and can manage and monitor all your hosts and services. M/Monit can start a service if it does not run, restart a service if it does not respond and suspend a service if it uses too much resources.

Monitor system attributes such as CPU, Load, Memory, Disk usage, Files, Directories and Filesystems for changes on all your hosts. Conditional rules can be set and if a value goes outside a defined scope, specific actions can be executed and notification sent.

Information is collected from monitored systems and stored in a database. Drill-down and filter functions exist to investigate collected data. Status and events from each monitored system are updated in real-time and displayed in charts, graphs and tables.

Benefits

M/Monit is a turn-key solution and requires very little configuration and no setup of third-party components.

Your computer systems will have a higher uptime as M/Monit can handle error conditions automatically, often without the need for human intervention.

M/Monit has a clean, simple and well designed user interface which scales well, if you manage 2 hosts or 1000+ hosts.

Source code with complete build system is available. Parts of the M/Monit system are also released as open-source code.

Cost-effective

A M/Monit license is a one-time payment (non-recurring cost) and the license does not expire.

The cost is minuscule compared to similar commercial systems and only a fraction of the cost as to the work hours required to setup and configure a comparable open-source system.

Technology

M/Monit is a modern, compact and scalable application server. Thread-pools and a non-blocking, event driven i/o architecture is used to ensure high performance. M/Monit runs on any POSIX system and use around 5-10 MB of RAM.

Database access is handled by a connection pool with support for MySQL, PostgreSQL and SQLite.

M/Monit for iPhone

Available for free from the iPhone App Store. [Click here to go to the App Store now.]

M/Monit for iPhone/iPod Touch are unsupported and provided as is.

System requirements

- M/Monit requires **Monit** as an agent. The Monit software must be version 5.2 or later and installed on all hosts M/Monit should monitor. Prebuilt Monit binaries can be downloaded from http://mmonit.com/monit/download/

- M/Monit runs on any **POSIX** system and is currently tested and available on Linux, FreeBSD, Solaris, Mac OS X and OpenBSD. If you need M/Monit on other systems, please let us know.

- **Memory** and **Disk** space. A minimum of 5 megabytes of RAM are required and around 20 MB of free disk space. You may need more RAM depending on how many processor threads the M/Monit server is started with, the number of login sessions that are used and the number of hosts monitored.

- **CPU** requirements. No special requirements. A single core CPU system should be able to provide enough performance to manage hundreds of Monit agents and hundreds of M/Monit web-app users.

- Accurate **time** keeping. M/Monit use the time of day for reporting and monitoring and it is recommended to investigate if your system clock has the correct time and set time synchronization facility on your system.

- **Random Device**. A random device is needed for creating universal unique and cryptographically strong HTTP (and SSL) Session identifiers. The Server will complain and exit if it cannot find `/dev/random` or `/dev/urandom` on the system.

- M/Monit **depends** only on the C-libraries installed on all POSIX systems.

M/Monit is **distributed** as a tar gzip archive with the following content:

- The mmonit program

- Dynamic shared libraries used by the mmonit program

- SQLite Database

- The M/Monit Web application

- Scripts for upgrading previous versions

- Source code and API documentation (Premium version)

Network communication requirements

M/Monit communicates with Monit agents on TCP port 2812. If there is a NAT or PAT (port translation) between M/Monit and Monit, you will need to setup host information in M/Monit so M/Monit can connect to Monit over the network. This can be specified in the admin/hosts page in the M/Monit web-app. Otherwise M/Monit will use the host information it receive from Monit when Monit automatically registered itself in M/Monit.

If communication from M/Monit to Monit agents are not available, M/Monit will not be able to manage services on a Monit Host, but M/Monit will still receive messages from Monit agents and display events and statistics.

M/Monit may need to connect to a SMTP and a Jabber server to send alert notifications. M/Monit may also need to communicate with a database server if so setup.

The M/Monit admin page displays a news feed from https://www.tildeslash.com/ Access to this feed is not required and if M/Monit cannot fetch the feed it is not displayed.

If you want to setup firewall rules for M/Monit, the default set of rules will be:

Host	Service	Port
Monit hosts and hosts with access to M/Monit	M/Monit web-interface	8080/TCP
M/Monit host	Monit agent	2812/TCP
M/Monit host	www.tildeslash.com	443/TCP
M/Monit host	SMTP server(s)	25, 465, 587/TCP
M/Monit host	Database server	port/TCP
M/Monit host	XMPP/Jabber server	5222, 5223/TCP

Installation

It is very easy to install M/Monit. All you need to do is download the tar gzip package from www.mmonit.com/download/ and unpack the tar.gz file. After unpacking, you will have a new directory called mmonit-2.4.

Now that you have installed M/Monit, it's time to launch it. Simply execute the mmonit program located in the bin directory. Then, point your Browser to the host where M/Monit is installed or *localhost* if it is running on the same machine as your Browser, for example: http://localhost:8080/ and login as user *admin* with password *swordfish*.

Once started, mmonit will run as a background process. To stop mmonit, use *mmonit stop*. To run mmonit in the foreground and in diagnostic mode, start mmonit with the -id options. In diagnostic mode, mmonit will print debug information to the console and also a short stack trace if any error occurs. Use ctrl+c to stop mmonit in diagnostic mode. To see all options for the program, use *mmonit -h*.

You can run mmonit as any user, including root. It is not necessary to create a standalone account to run mmonit.

The directory doc/startup contains general startup scripts you can use to start mmonit at boot time on your system.

Resources

Website

M/Monit's website has a lot of information about M/Monit and Monit. It features up-to- date information about the application, including new and updated features, development news, and tutorials. The website also sports a wiki with more information and user submitted content.

Mailing List

You can subscribe to M/Monit's mailing list to be the first to hear about new releases and important information about M/Monit. The mailing list is read-only with very low traffic.

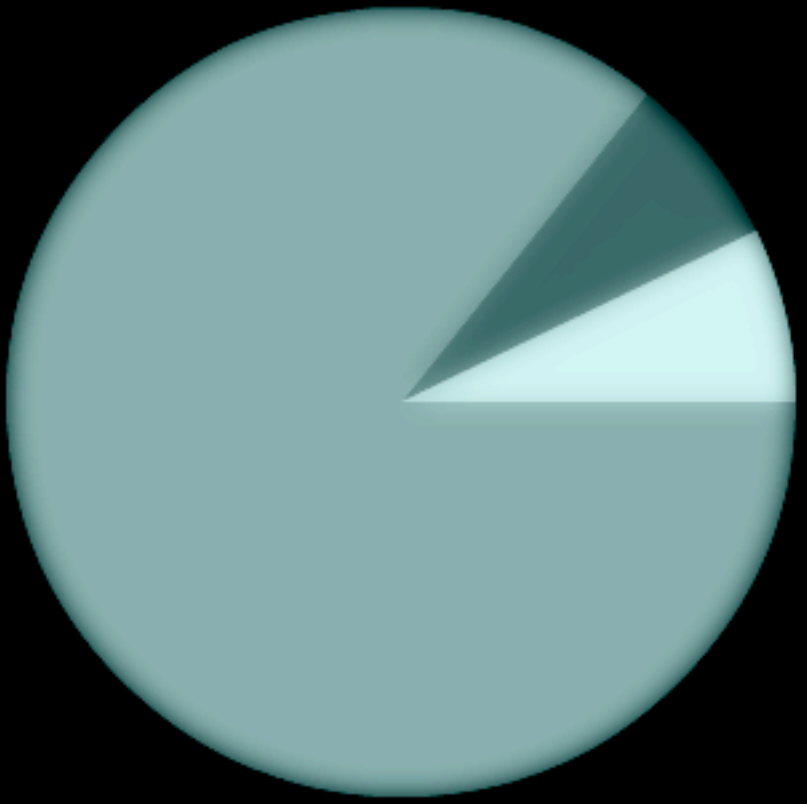
Feedback and Bug Reporting

Though we invest a lot of time and hard work making sure Monit and M/Monit are high-quality applications, bugs may still sometimes find their way into the application. Use the contact information at mmonit.com/contact/ to let us know about any bugs or feature requests you have.

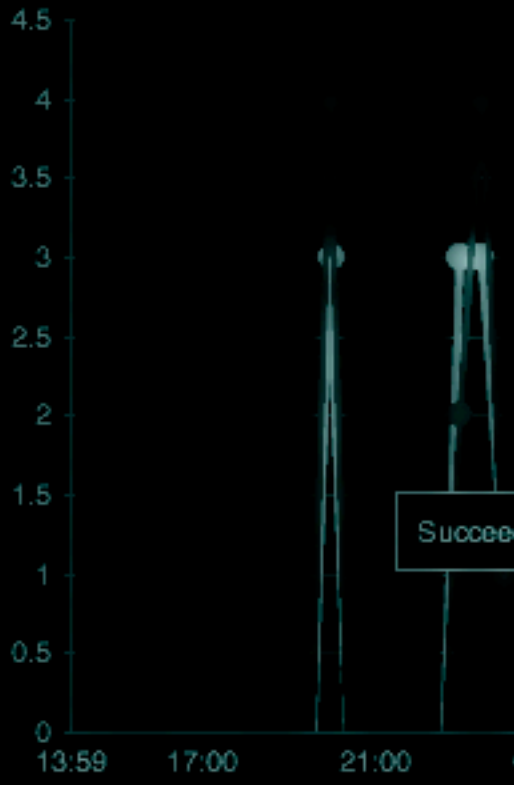
You can also contact us directly at info@mmonit.com

Latest status

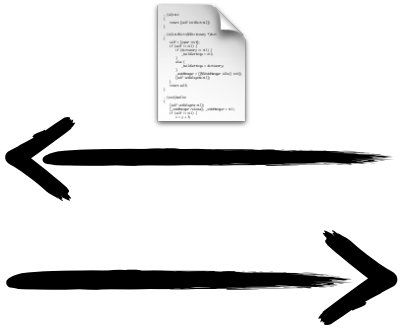
Hosts Status



Events in last 24-hours



Architecture overview

M/Monit  Monit

Monit is a small, powerful monitoring program that runs on each host monitored by M/Monit. With regular intervals, Monit will send a message to M/Monit with the status of the host it is running on. If a service fails or Monit has to perform an action to fix a problem, an event message is sent to M/Monit at once. Both status and event messages are stored in M/Monit's database. Upon receiving an event message from Monit, M/Monit will consult its rule-set and perform alert notification if a rule match.

From M/Monit, you can start, stop and restart services on any of your hosts. M/Monit will delegate to Monit to perform the requested operation. In addition, M/Monit will use Monit to display a host's detailed status information in real-time.

Configuration

Before we take a look at the M/Monit web-application let's take a quick look at M/Monit and Monit configuration. As we mentioned earlier, Monit must be configured in order to send event and status information to M/Monit. In the future we plan to add zero-conf capabilities to M/Monit and Monit, but for now you will need to add a few lines to Monit's configuration file. To learn more about Monit and Monit configuration, please consult the documentation and configuration examples at Monit's website.

To setup communication between Monit and M/Monit, add the following statements to the top of each Monit control file, `monitrc`:

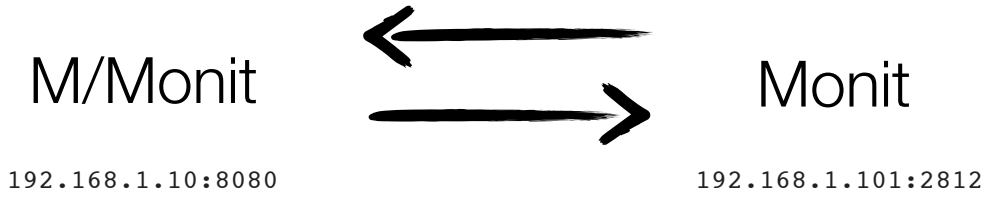
```
1.   set eventqueue basedir <path> [slots <number>]
2.   set mmonit http://<user>:<password>@<host>:<port>/collector
3.   set httpd port 2812 and use address <monit-host>
4.       allow localhost
5.       allow <M/Monit-host>
6.       allow username:password
```

The *set eventqueue* statement in line 1 is optional, but recommended. It allows Monit to store event messages if connection to M/Monit should temporarily be unavailable and retry delivery later. This way, no events will be lost. The slots option can be used to set a limit on how many events can be stored so the queue won't grow without limits if M/Monit is not available. The size of a queued message is small (ca. 200 bytes) so the space requirements for, let's say 1000 queued events is only 200kB.

The *set mmonit* statement in line 2 specify the URL to be used by Monit for sending messages to M/Monit. The M/Monit URL is protected, and a username and a password are required to post messages to M/Monit. Use the username and password of any valid user in M/Monit. For instance, the default user, "monit" with password "monit". The host and port in the URL, specify respectively the IP address of the machine running M/Monit and the port on which M/Monit is listening.

The *set httpd* statement starting on line 3 allow M/Monit to connect to Monit. Specify the IP-address or host name of the host running Monit and the IP-address of the host running M/Monit, this should be the same address as specified in the *set mmonit* statement mentioned above.

Finally, in line 6, specify a Basic Authentication username and password M/Monit should use to login to Monit.



Assume we have the machines above where M/Monit runs on 192.168.1.10 and listen on port 8080. And Monit runs on 192.168.1.101 and listen on port 2812. You should then add the following to your *monitrc* configuration on 192.168.1.101:

```
set eventqueue basedir /var/monit/ slots 1000
set mmonit http://monit:monit@192.168.1.10:8080/collector
set httpd port 2812 and use address 192.168.1.101
    allow localhost
    allow 192.168.1.10
    allow admin:secret
```

For extra security you can configure M/Monit and Monit to use **SSL**. In the example below we demonstrate how to setup the two-way communication between M/Monit and Monit to use SSL.

The only differences from the above example is that the *set mmonit* statement now uses https instead of http and M/Monit's SSL port. In addition we add *SSL enable*, *pemfile* and *allowselfcertification* to the *set httpd* statement to enable SSL in Monit. For more information on how to setup Monit to use SSL please see the Monit wiki. How to setup *M/Monit* to use SSL will be addressed shortly.

```
set eventqueue basedir /var/monit/ slots 1000
set mmonit https://monit:monit@192.168.1.10:8443/collector
set httpd port 2812 and use address 192.168.1.101
    SSL enable pemfile /path-to/monit.pem
    allowselfcertification
    allow localhost
    allow 192.168.1.10
    allow admin:secret
```

We recommend to setup Monit and M/Monit to use SSL if communication between M/Monit and Monit goes through the cloud and not over a local network.

After you have changed Monit's configuration, you will need to reload or restart Monit. Monit should now start sending messages to M/Monit and automatically register itself, that is, create a host entry for itself in M/Monit's database. At this point, you should be able to see the Monit host in M/Monit's web interface.

Monit ID

Each Monit instance is identified by a unique id, stored in the file, $HOME/.monit.id on the host running Monit. $HOME is the home directory of the user running Monit. This file is automatically created at startup by Monit if it does not exist. Care should be taken not to delete or change this file as the ID is used to pair messages from Monit with a host entry in M/Monit.

If you want to place the id-file in another location other than the default, move the id file to its new location and add a *set idfile* statement in *.monitrc* to specify the new location of the Monit id file.

Configuring M/Monit

Configuring M/Monit is usually not necessary, but you may want to change the **port** number M/Monit use if you already have a server running on port 8080. In addition, you may want to setup M/Monit to use **SSL** or use another **database** other than the default built-in SQLite database.

M/Monit configuration files

The configuration file for the M/Monit server can be found in the conf directory and is called *server.xml*. A detailed discussion about server.xml and its directives can be found in the appendix. The other configuration file in the conf directory is *web.xml*. This file specify mime-mappings for the application and need not be changed. The M/Monit web-application also use a *web.xml* configuration file in `docroot/WEB-INF/` This file should not be changed and comes preconfigured for the application.

How to change the port number M/Monit listen on?

Change the port attribute in the Connector element if you need M/Monit to listen on a port other than 8080. For example to setup M/Monit to listen on port 8888 on all interfaces:

```
<Connector address="*" port="8888" processors="10" />
```

By default M/Monit binds to all network interfaces and can be reached via any address on the server it is running. If you need M/Monit to only bind to a specific interface, change the address attribute in the Connector element. For example, to setup M/Monit to only bind to 192.168.1.10 use:

```
<Connector address="192.168.1.10" port="8080" processors="10" />
```

How to setup M/Monit to use SSL?

Set the *secure* attribute in the Connector element or uncomment the SSL Connector in server.xml.

```
<Connector address="*" port="8443" processors="10" secure="true"/>
```

Then specify the IP-address and the SSL certificate to be used for the default host:

```
<Host name="localhost" appBase="." address="192.168.1.10" certificate="conf/mmonit.pem" >
```

That is it. Just replace the IP-address in the example above with the IP-address of your own host. Restart mmonit and connect securely using https://192.168.1.10:8443/ or https://yourhostname:8443/

M/Monit comes with a self-signed certificate in the conf directory called mmonit.pem which you can use for testing. In production you should use your own certificate as the private key in mmonit.pem is no secret. Note that a self-signed certificate will generate a warning in your browser, saying that the certificate is invalid. This does not mean that the connection won't be secure, just that the browser cannot validate the identity of the server. In our case we can safely ignore this warning and click continue in the browser.

If you already have a SSL certificate for your server and want to use this instead, simply replace the content of mmonit.pem with your server private key, server certificate and the certificate of the Certificate Authority that signed the server certificate, in that order. You *must* also change the name of the <Host> to match the Common Name in the certificate. For instance, we have bought a SSL certificate for our server www.tildeslash.com and in our case we would use this configuration:

```
<Host name="www.tildeslash.com" appBase="." address="62.109.39.247"
      certificate="conf/mmonit.pem" >
```

The name attribute should be the DNS A-record name of the server and the address, the IP-address www.tildeslash.com points to in DNS. We must also remember to change the defaultHost attribute in the <Engine> element to our host name. In our case we would use www.tildeslash.com and specify the Engine element like so:

```
<Engine name="mmonit" defaultHost="www.tildeslash.com" fileCache="10MB">
```

You can run M/Monit with both a SSL connector and a non-SSL connector. This is useful if you need M/Monit to listen on more interfaces. M/Monit can only use one interface with SSL because the certificate will only match one host name, while with the non-SSL connector you can use all interfaces on your host. Connections from the outside could connect using SSL, while hosts behind your firewall can connect via internal IP-addresses/interfaces using the non-SSL connector if you wish.

How to setup M/Monit to use MySQL or PostgreSQL?

M/Monit comes bundled and configured with SQLite as its database system. No extra setup is required. We use SQLite with our own M/Monit installation and we like the simplicity of it, but we only monitor a few hosts though. If you plan to use M/Monit to monitor more than, say 20-30 hosts, you may want to use MySQL or PostgreSQL instead as these database systems are faster and scale much better. If in doubt, start with SQLite. If you later should want to switch, you can use the migrate script in the *upgrade* directory to move your SQLite data over to MySQL or PostgreSQL.

Setting up M/Monit to use either MySQL or PostgreSQL is a simple two step process:

1. Create the M/Monit database. The database schemas with recipes for creating the database can be found in the db directory in the M/Monit home directory.
2. Configure M/Monit; edit the M/Monit configuration file server.xml and replace the default SQLite Realm element with either this one for MySQL:

```
<Realm url="mysql://mmonit:mmonit@127.0.0.1:3306/mmonit"
       minConnections="5"
       maxConnections="25"
       reapConnections="300" />
```

   Or this one for PostgreSQL:

```
<Realm url="postgresql://mmonit:mmonit@127.0.0.1:5432/mmonit"
       minConnections="5"
       maxConnections="25"
       reapConnections="300" />
```

   Change username, password, host and port number in the connection URL as required.

M/Monit is known to work out of the box with PostgreSQL 7.4 - 8.x and with MySQL 5.x.

If you have a later database version and experience problems, you can download and build the libzdb connection pool library used by M/Monit with your database version. Simply replace the libzdb shared library in mmonit/lib/libzdb.x with your own built version. Libzdb is open source and can be downloaded from libzdb's website.

How to increase the login session timeout in M/Monit?

When you log in to the M/Monit web-application a new session is created. The login session will timeout if there has been no activity for a specific time and users must log in again after a timeout. The session timeout value is specified in the <Context> element as number of seconds in the *sessionTimeout* attribute. The default timeout is 30 minutes, that is, 1800 seconds. To increase this, for instance to one hour, use *sessionTimeout="3600".*

If you click the “Remember me” checkbox in the login screen, a persistent session is created instead. This session will timeout after 3 months and will be stored in the M/Monit database. All sessions in current use will survive a mmonit server restart.

How to install a license key?

M/Monit comes with an evaluation license which will expire after 30 days. You can buy a license online which does not expire. The license key will be sent in an email. Replace the existing license element in server.xml with your new key and restart M/Monit. If you go to the M/Monit admin page you will see more information about your license and how to contact us if you need support.

A First Look at M/Monit

Dashboard click the logo to go directly to the dashboard

Logout from M/Monit

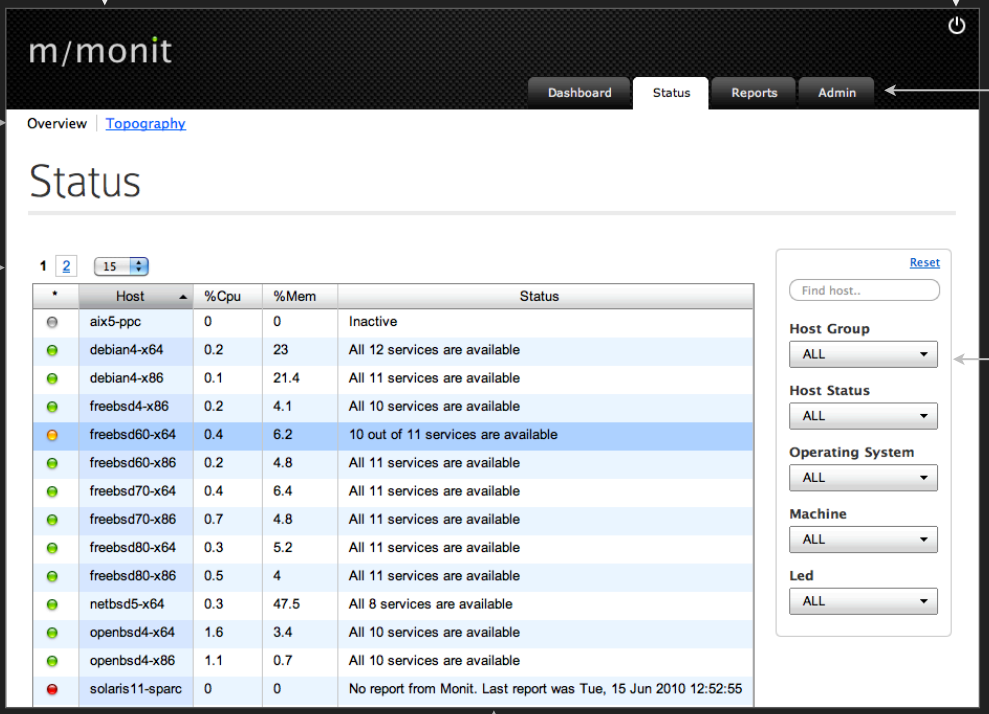
Sub-menu displayed in some pages, such as Status and Admin

Navigation tabs Access Dashboard, Status, Reports and Admin

Paginator If there are more than a certain number of rows, the table displays a subset of available rows and the paginator can be used to navigate remaining rows.

The drop-down menu can be used to select number of rows to display in the table.

Drill-down menu Use the drill-down menu to drill down into data displayed in the table



Tables are used many places in M/Monit to display data. Click a column header to sort rows in ascending or descending order. Move the mouse cursor over the table and the active row under the cursor is marked in blue. To see entry details, click the row.

Login to M/Monit

After starting the mmonit program, point your Browser to http://localhost:8080/

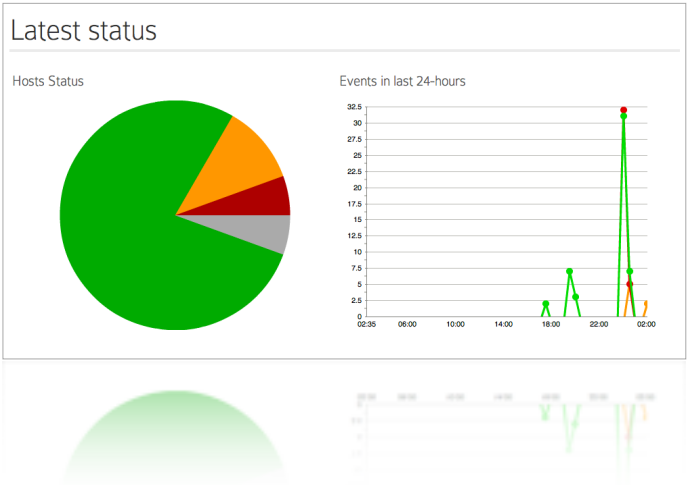
M/Monit comes pre-installed with two user accounts you can use the first time to login:

User	Password
admin	swordfish
monit	monit

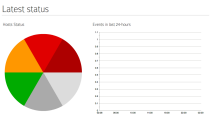
The admin user is a member of the administrator role and has access to every page and functionality in the web-app. The monit user is a regular user and cannot access the admin pages nor manage services in the status page.

Enter admin and password swordfish and click the nice green login button. The *Remember me* checkbox can be used for persistent login, if checked, a login session is created with timeout set to 3 months and the session is also stored in the database.

Dashboard



The dashboard is the first page you will see after login. This page provides a quick overview of the status of all hosts monitored by M/Monit and events coming in from the last 24 hours.



The first time you log in to M/Monit, the dashboard will look like the screenshot to the left. Since no Monit hosts have been registered, the pie chart will show all colors and the events chart will be empty.

Later, when Monit is setup and start to report in, this page is going to be useful for getting a quick status overview of all your hosts. The charts refresh themselves automatically each minute (the refresh rate can be set in the users page under admin). The Host Status pie chart display the status of hosts registered in M/Monit and if all hosts and services are online, the pie chart should be all green.

Here are the possible pie chart colors and their meaning:

Host is offline

Host with all services offline

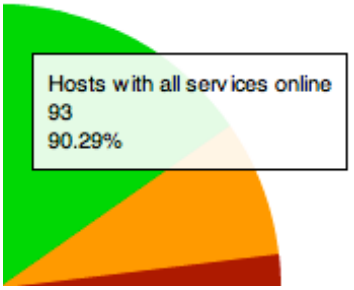
Host with some services offline or in unmonitored state

Host with all services online

Ignored host

Inactive host

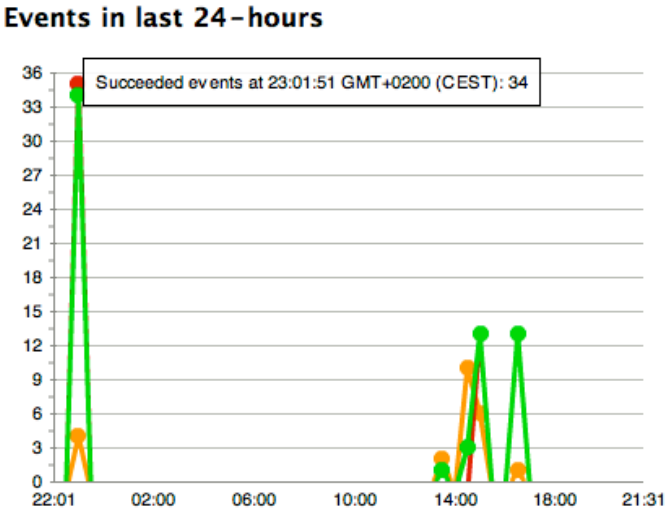
As indicated in the picture to the right, you can hoover the mouse cursor over the pie chart to see a popup window with a description of each segment. A click anywhere in the pie chart will open the Status page.



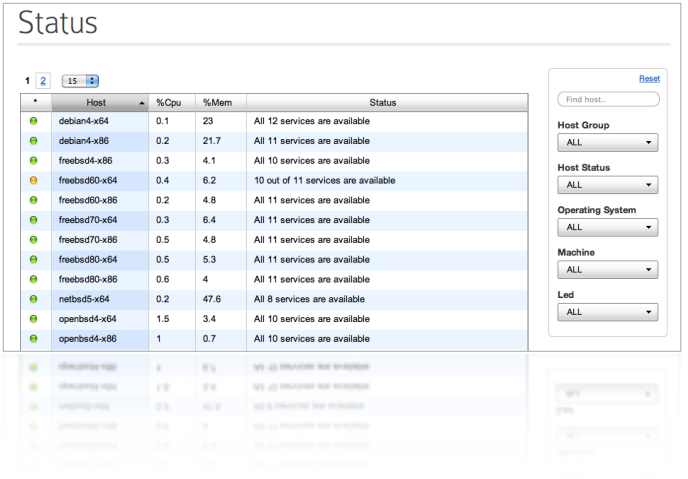
The *Events in last 24-hours* chart, plot events coming in from Monit over the last 24-hours. Events are plotted from right to left. That is, the most recent event are plotted rightmost in the chart. The chart plot the following colorful events:

green	Success, a previous reported error was fixed
red	Failed, a service failed
orange	Changed, such events are sent when Monit start/stop, when an action is performed, or when a service fire a change event. For instance if a file checksum or a process pid changes.

If you hoover the mouse pointer over a plot-point, a popup window is displayed with event information as indicated in the picture to the right. Click on a plot-point to open the events log.



Status



The Status page shows the status of each monitored host based on periodic reports from Monit. A status table row have the following columns,

*	Host	%Cpu	%Mem	Status	Events
●	debian4-x64	0.2	23.4	All 12 services are available	1017

The first column display a LED representing the host's error state:

- Error. Either the host did not report in or all services are down
- Warning. At least one service is down or in unmonitored mode.
- OK. All services are up and running
- Host is in Ignored or in Inactive state.

The next columns display the host's descriptive name, the cpu% and the memory% usage on the host. The last two columns display services status and the number of events registered on the host. Clicking on the link in the events column will bring up the events log with drill-down filters preset for the host.

Hint: You can sort the table by clicking on a column header. Sorting the status table on the LED column is useful to quickly see hosts with errors or warnings. The drill-down menu can be used to filter the table on various criteria. *Note* that values in the drill-down menu are persistent across page reload. Use the reset link to reset the drill-down menu and display all hosts.

By default, the table auto-refresh itself each minute. This means you can sit back and watch the table without hitting reload. To make this chart more interesting to sit back and look at; tweak down Monit's poll time to 5 seconds and set the (table) refresh rate to 5 second in admin/users.

Click a table row to display detailed hosts status, M/Monit will connect to Monit and display the host's current status in *real time*. Access to Monit require that Basic Authentication credentials for Monit has been set in M/Monit. When Monit register itself the first time it send its credentials as part of the registration process. You may configure Monit *not* to send credentials at registration. If this is the case, the first time you click on a host you will see the following error (Invalid username or password):

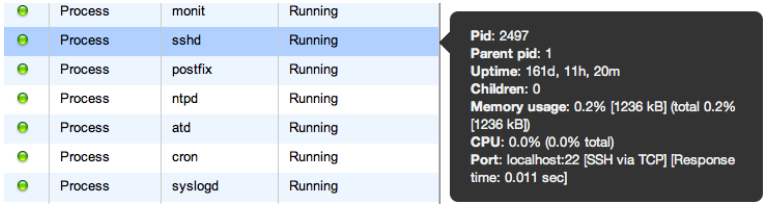
Cannot connect to Monit -- Invalid username or password
Please check latest events for the host and host configuration.

To resolve this problem, click on the host configuration link and set the credentials for the host. That is, enter Monit's username and password as it was specified in Monit's control file, *monitrc*, on the host.

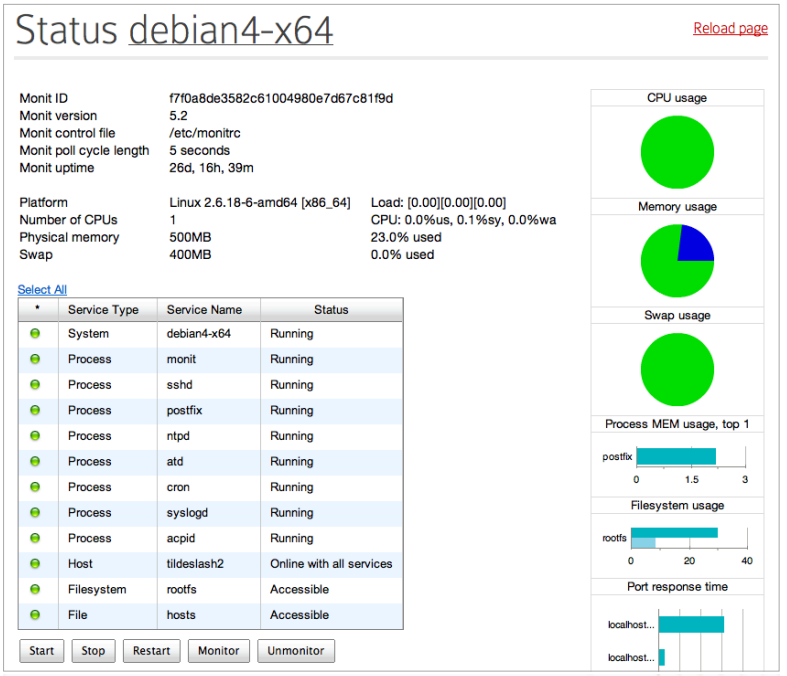
Detailed host status

This page display detailed host status. Status and information are in real-time. The table list all services running on the host, that is, all services monitored by Monit.

If you move the mouse cursor over a table row, a pop-out panel is displayed with detailed information about that service:



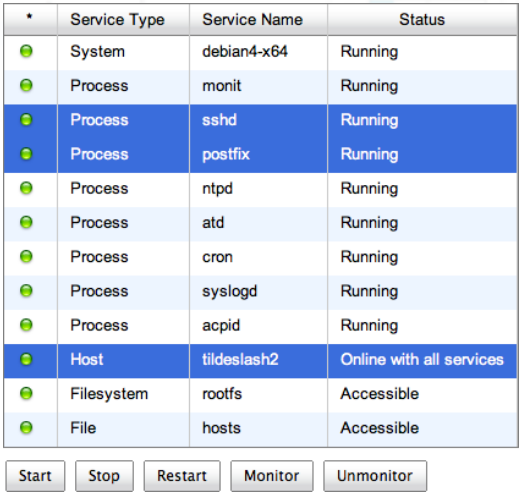
The charts on the righthand side display various host attributes; Such as CPU, MEM and disk usage. If you move the mouse cursor over a chart, a popup-window is displayed with values.



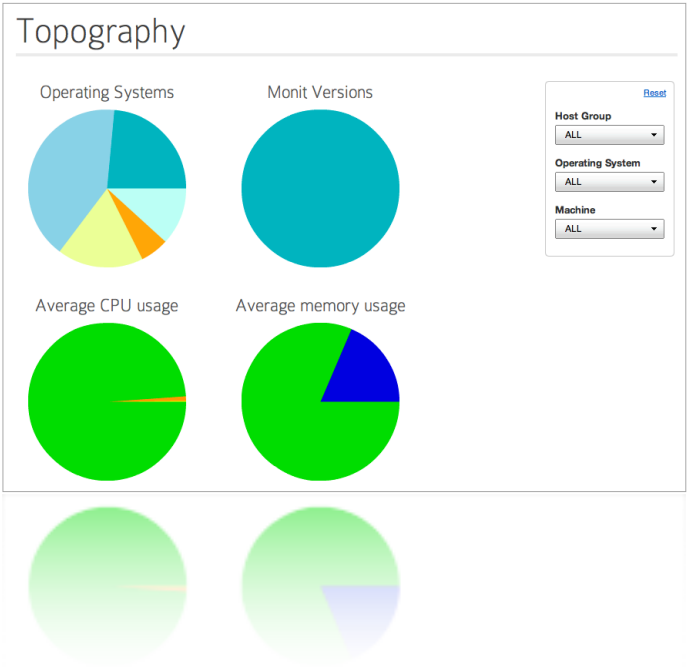
Service actions

Select a service by clicking a table row. Multiple rows can be selected by holding down modifier keys such as CTRL or SHIFT or you can select all rows by clicking the [Select All] link. To unselect rows, press *ESC*. (When a row is selected, the black pop-out panel with service information is not displayed. Simply unselect rows (ESC) to display this panel again).

Once a row has been selected you can use one of the action buttons to start, stop, restart, monitor or unmonitor the service(s) on the host. Note that start, stop and restart require that the service has a start and a stop program registered in Monit. All services can be set in a monitor or unmonitored state.



Topography



The topography page display aggregated statistics from all hosts. Hoover the mouse cursor over a chart to see a popup-window with detailed information. Use the drill-down menu on the right to drill-down on various criteria, for example:

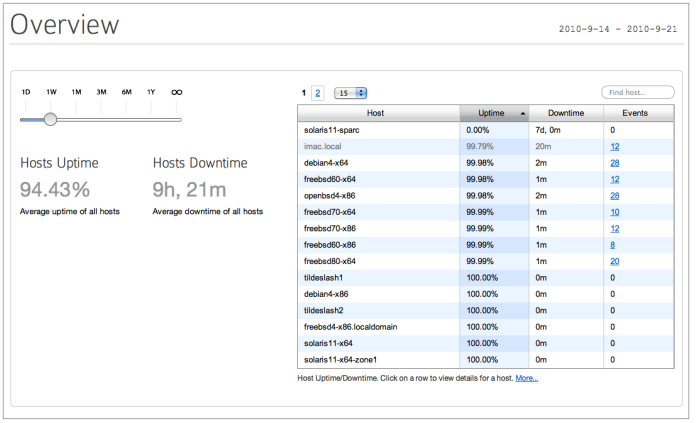
- Check average memory utilization of all hosts in a host group
- Determine CPU and memory utilization based on machine architecture. How many CPU resources are available for e.g. X86 servers
- Determine CPU and Memory utilization on Linux-based servers

This provides for a very basic overview for resource planning; upgrades, locate free resources for service addition, watch key performance indicators, etc.

The following topography charts are currently available:

- **Operating systems** Shows the percentage of each operating systems in use. This provides for a quick overview of how many hosts are running which OS.
- **Monit versions** Show which Monit version is in use. Useful for planning Monit upgrades.
- **Average CPU usage** Average CPU usage across all active hosts.
- **Average memory usage** Average memory usage across all active hosts.

Reports

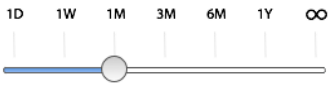


The overview report show uptime and downtime of all Hosts monitored by M/Monit. Hosts with downtime include hosts that are actually down, the network from M/Monit to the host is down or hosts where Monit was not gracefully stopped.

If Monit does not send a status message within a specific time (3 Monit poll cycles) M/Monit will assume the host is down and raise a status failed event. This event is associated with the "monit" service and can be seen in the events log and in the status page as "No report from Monit. Last report was...". When Monit start sending status messages again, M/Monit will raise a status succeed event. This can be seen in the events log as "Monit status report received successfully".

Host uptime is based on both succeeded and failed status events. Resolution is in minutes and the combined downtime for a host must accumulate to at least one minute for the host to appear with downtime in the table.

Selecting a date range



Click and drag the slider button or simply click an interval indicator to select a date range (right arrow key can also be used). The range's *from* and *to* date can be seen at the top right of the reports page. The date format is year-month-day. The from date is the first and today's date the second.

2010-8-20 - 2010-9-20

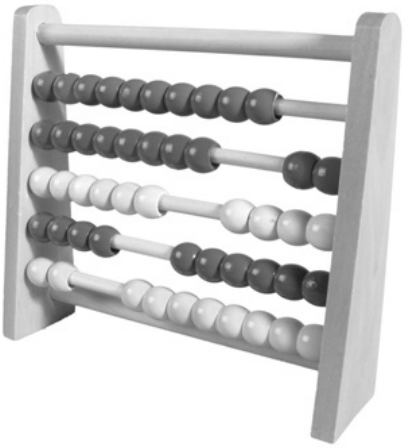
Choosing **1D** sets the range from midnight today and until now. **1W** sets the range from 7 days in the past and until now, **1M** is one month with blast from the past and so on. The **∞** range indicator sets the range from the oldest registered status event in M/Monit and until now.

Average uptime and downtime

The average uptime and downtime of *all* hosts are displayed to the left of the table. Inactive or ignored hosts are not included in the average unless they have downtime.

$$\frac{1}{n} \cdot \sum_{i=1}^{n} u_i$$

How uptime/downtime is calculated:



1. If a matching failed and succeeded status event is found within the selected date range, the difference in time between the two events is counted as downtime.

2. For open ended failed status events, that is; failed events without a matching succeed event, the downtime is counted as the difference between *now* and the failed status event's timestamp.

3. If a host has not reported within the selected range, the host is assumed to be down if and only if the host is active and its last updated timestamp is older than the selected range's from-date. In this case the difference in time between now and the range's from-date is added to the host's downtime. I.e. the host is down the full range.

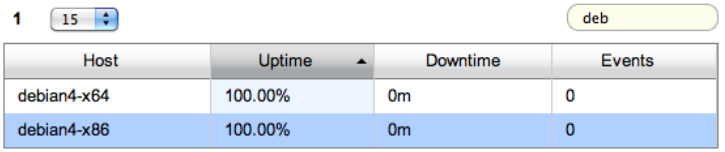
4. Inactive and ignored hosts are not included in the total average numbers unless they have downtime. Their uptime/downtime is set to zero if they have no downtime within the range. You can see inactive and ignored hosts in the table with a grey name.

Events

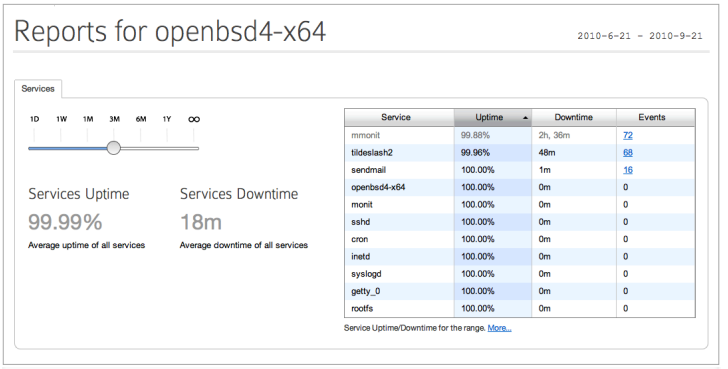
The events column show the number of status events for the host *within* the selected range. Clicking the number brings up the events log with drill-down filters preset to only show events for the host within the selected range and sorted on date in ascending order.

Host filter

If you monitor more than 15 hosts, a paginator and a host search field are displayed at the top of the table. The search field can be used to focus on selected hosts only. The table remains filtered even if the date range changes. To reset, remove all text in the search field.



Service Report



If you click a host entry in the Host uptime report, the service report is displayed with uptime and downtime for services monitored by Monit on that host.

As with the Host report, the uptime is based on both succeeded and failed events. Resolution is in minutes and the combined downtime for a service must accumulate to at least one minute for the service to appear with downtime in the table.

The uptime and downtime of services are calculated similar to the Host report;

1. If a matching failed and succeeded event is found within the selected range, the difference in time between the two events is counted as service downtime.
2. For open ended failed events, that is; failed events without a matching succeed event, the service downtime is counted as the difference between now and the failed event's timestamp.

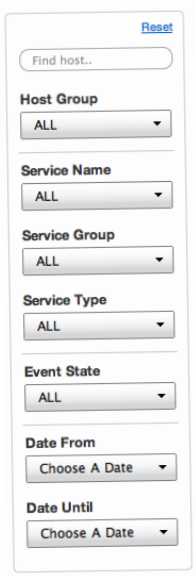
β The report work around the fact that Monit does not maintain services (failed) state over a Monit restart/reload nor during service un-monitoring. This may add some inaccuracy to the computed downtime, but in practice it should work well. However, because of these exceptions, this report is a *beta* and the numbers should be seen as more informative than authoritative.

Grayed out entries

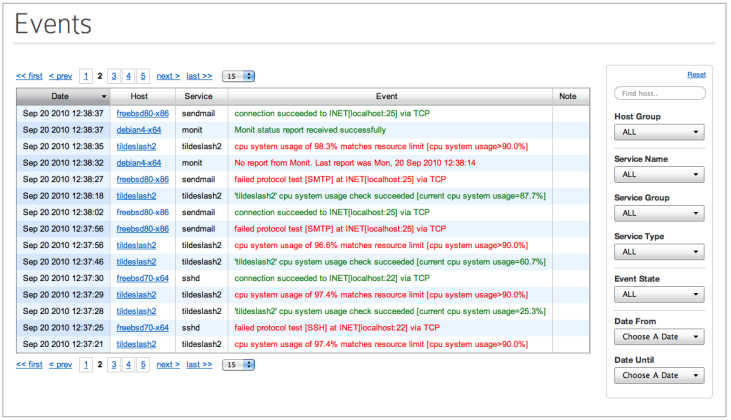
1. If the Host is inactive or ignored all services in the table are grey.
2. Services that are in an unmonitored state are marked as grey. Their uptime/downtime is set to zero unless they have actual downtime within the selected date range. Unmonitored services are not included in the total average numbers unless they have downtime.
3. Services that once existed in your Monit control file (.monitrc) but since has been removed may show up if the selected range is wide enough. These historical services are grayed out.

The Events column show number of events for the service *within* the selected range. Clicking the number brings up the events log with drill-down filters preset to only show events for the service and within the selected range.

Events



The Events log can be used to browse all events reported by Monit and stored in M/Monit's database. The log is initially sorted on date in descending order. I.e. the latest events are listed first. Internally, M/Monit buffer new events in an event queue and flush new events to the log each 5 seconds. The events log will grow over time as no events are removed.



Use the drill-down menu to filter the log on various criteria. One useful criteria is date. For instance, to only show events for a certain date or within a range by using the Date From and Date Until calendar buttons.

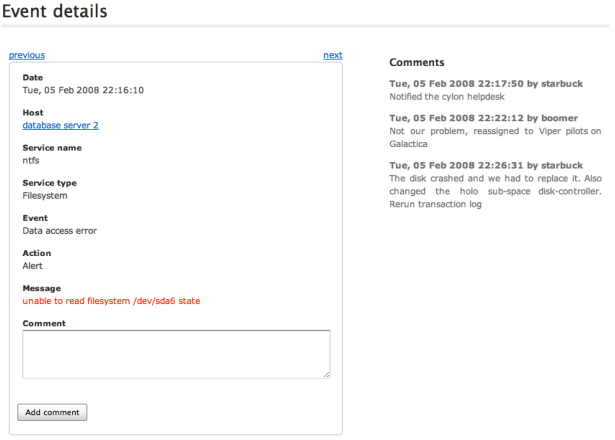
Hint: If you click the month in the calendar you can quickly navigate by year or month.

If you enter a Host name in the drill-down search field, only events for that host are displayed. Service Name in the drill-down menu will only display services for the Host and the Service Group will only display Service Groups, if any, for the Host. *Note* that values set in the drill-down menu are persistent across page loads. Use the reset link to reset all drill-down values.

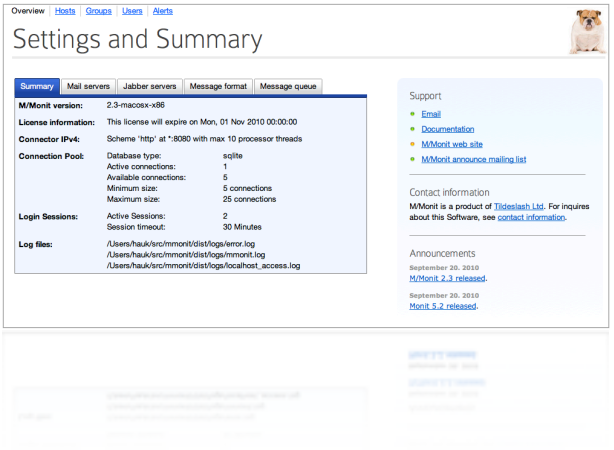
Event details

Click an event row to show details for the event. In addition to viewing a few more details you can also add comments. Events with comments are marked in the log with this icon 

Comments can be used for tracking events handling, explain the root cause of the problem, consequences and serve as knowledge base for problem handling.

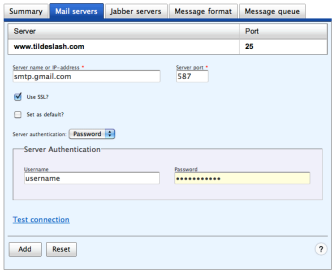


Admin



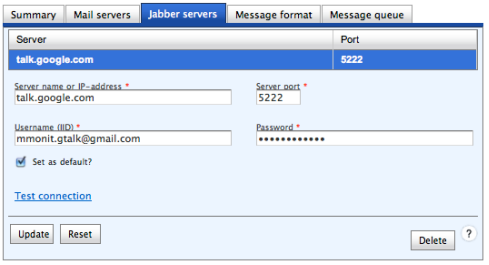
The Admin function is split into several pages accessible from a submenu. Only users in the admin role has access to these pages. The first page display systems information, support and contact information.

If M/Monit is used with an evaluation license, the expiration date for the license is displayed in the Settings and Summary tab, otherwise, your license information is displayed, such as the license serial number and license owner. This information should be provide when requesting support so we can prioritize your request.



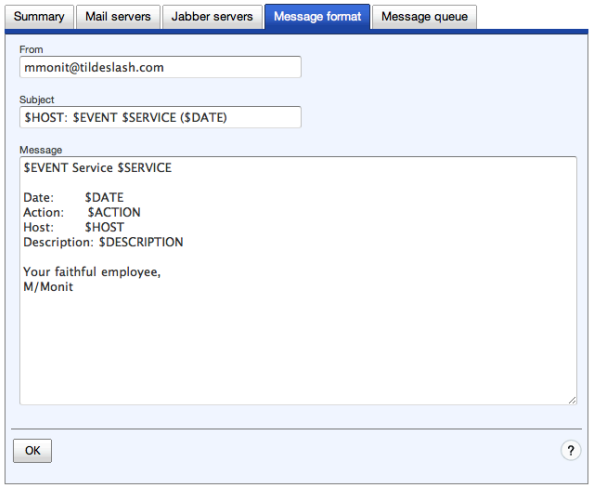
Click the **Mail servers** tab to specify SMTP servers M/Monit should use for alert notification. To specify a new server, fill in required fields and click Add. If the Add button is not displayed, click Reset first. To edit an existing server, select the server from the table, edit fields and click update. When sending an alert message, M/Monit will start with the default Mail Server and if not available, try the next Mail Servers in the list until the message is sent. If the server requires authentication, select password from the drop-down box and fill in username and password. If the SSL checkbox is checked, M/Monit will use STARTTLS if the mail server port number is 25 or 587, otherwise SMTPS is used. The Test connection link can be used to verify the current server. This will test if M/Monit can connect to the mail server and, if specified, test if authentication and SSL works.

M/Monit can also send alerts as instant messages (IM) using the **Jabber** protocol. The most famous IM application using this protocol is Google Talk (GTalk). If you are on OS X, iChat works fine with GTalk and can be used to display IM alerts from M/Monit. If you plan to use GTalk, the server name should be, *talk.google.com*, the port number 5222, the username should be a gmail.com address and the password the gmail password.



Note that the user you specify here should be different from the one you will use in your Jabber client. Otherwise the Jabber server will think that you are sending messages to yourself and not send the alert message to your client. We recommend that you create a new dedicated gmail account to be used by M/Monit only.

The first time M/Monit sends an IM message you will be asked by your client to add the M/Monit user to your buddy list. You should accept this request since M/Monit can only send alerts to your client if you have the M/Monit user in the buddy list. Alternatively, you can pair manually using your Jabber client to add the M/Monit user to your buddy list.



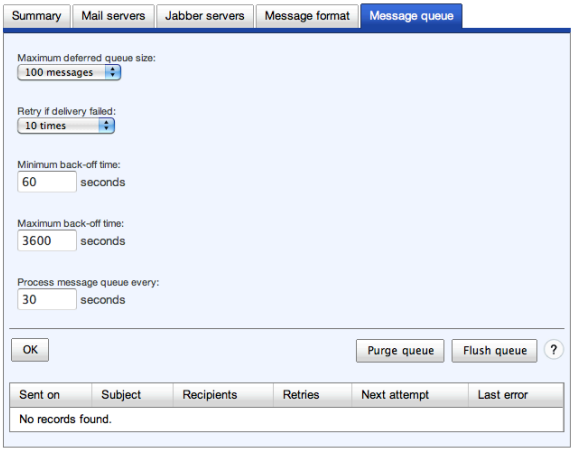
Message format specify the template M/Monit should use for alert messages, both for mail and IM. Substitution variables can be used and will be expanded when the message is sent. The following substitution variables can be used in the message:

Variable	Description
$EVENT	A short string describing the event that occurred
$SERVICE	The name of the service generating the event
$DATE	The date and time the event occurred on the Host
$HOST	The name of the Host the event originated from or if used in the From field, the name of the M/Monit host
$ACTION	The name of the action which was performed by Monit
$DESCRIPTION	A short description of the event condition. I.e. why the event was sent

Alert messages are added to the M/Monit **message queue** and the queue is processed every 30 seconds by a transmit thread which send all queued messages.

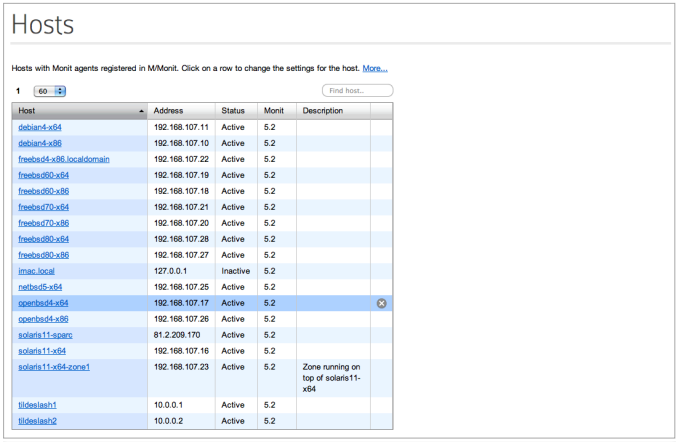
M/Monit will queue messages in a deferred queue if messages cannot be sent and resend should be attempted later. Messages are only put in the queue if a temporary error occurred, such as, if no servers are online or if the server returned a temporary error. If the server returns a permanent error, messages are dropped and not added to the queue. If the queue has reached maximum size, M/Monit will drop new failed messages. Even if the deferred queue is full, M/Monit will try to deliver new messages at least once.

The retry count specify how many times M/Monit should retry sending a message before giving up and dropping the message. M/Monit will try to resend a message using an exponential back-off strategy; starting with minimum back-off time and increasing the retry time with every failed attempt up to maximum back-off time.



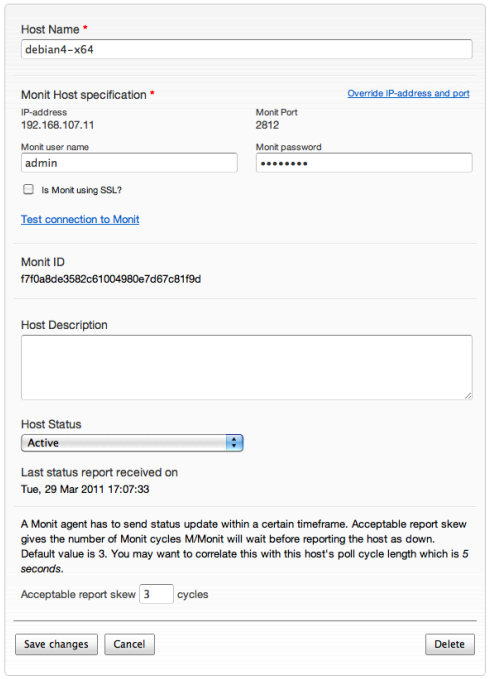
The content of the deferred queue is shown at the bottom of the message queue panel. Under normal circumstance this table should be empty. Buttons are available to purge the deferred queue or flush all queues. It is also possible to set unlimited retries and the size of the deferred queue in this panel.

Hosts



The hosts page list all hosts registered and managed by M/Monit. Monit will *automatically* create a new Host entry the first time it reports in to M/Monit. To make Monit report to M/Monit, use the set mmonit statement in your *.monitrc* file. After Monit has registered itself in M/Monit you can edit the Host by selecting the Host in the table.

The “Find Host” search field above the Host table can be used to quickly filter the table on a host name. To edit a host, click the host’s table row.



The first time Monit register a Host, the Host Name is set to the DNS name of the host or its IP-address if the host does not have a DNS name. This name can later be changed to something more descriptive, such as database server 1 or Babylon 5. The descriptive Host Name is used in various places in M/Monit to refer to the Host.

Monit Host specification fields are used to specify connection from and to Monit. IP-address is the Monit host's address. This value is set by Monit and cannot be changed. If you need to override the value, click the Override IP-address and port link which will display another pair of IP-address/port fields for specifying outgoing connections to Monit from M/Monit. You may enter a host name instead of an IP-address in the outbound IP-address field, M/Monit will use DNS to resolve the host name to an IP-address when it connects to Monit. Monit Port number is the HTTP port Monit is setup with on the host.

The Monit user name and Monit password fields specify a username and a password to be used for HTTP Basic Authentication with Monit. By default, Monit will register credentials the first time it reports to M/Monit so you do *not* have to enter the username and password manually. This automatic registration of credentials can be turned off in your *.monitrc* file though it is recommended to keep it otherwise you must manually enter the credentials for each host. If security is a concern, setup Monit and M/Monit to communicate over SSL and check the SSL-checkbox.

Monit ID

Each Monit instance is identified by a unique ID, stored in the file, $HOME/.monit.id *on the host running Monit*. $HOME is the home directory of the user running Monit. This file is automatically created at startup by Monit if it does not exist. Care should be taken not to delete or change this file as the ID is used to pair messages from Monit with a host entry in M/Monit. If you want to place the id-file in another location other than the default, move the id file to its new location and add a *set idfile* statement in *.monitrc* and specify the location of the Monit id file.

A new host is automatically created in M/Monit by a status message from Monit if and only if the ID embedded in the message is new and unique, otherwise M/Monit will use the ID to find the host in its database and update the host status.

Host Status

A Host can be in one of three states:

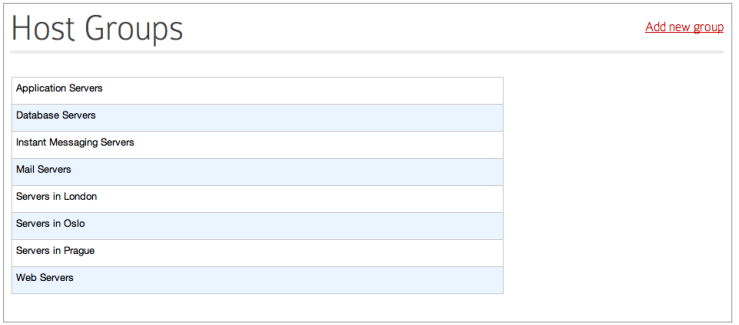
- **Active** When Monit start, it activates itself automatically in M/Monit by sending a start event message. M/Monit expect periodic status reports from active hosts. If a host does not send a status report within a certain time frame it is marked as down and an alert is generated by M/Monit.
- **Inactive** If Monit was stopped correctly, it deactivates itself automatically by sending a stop event message to M/Monit.
- **Ignored** The Ignored state can be used to temporarily suppress all events from Monit. For example, if you are going to perform maintenance work on a host, and you expect Monit to issue lots of alerts, set the Host in Ignored state to ignore events sent by Monit for the duration of the work.

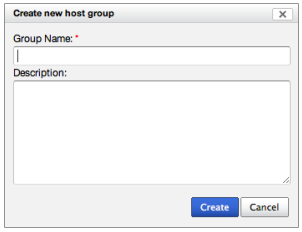
Report skew

Monit has to send status update within a certain timeframe. Acceptable report skew specify the number of Monit cycles M/Monit will wait before reporting the Host as down. The default value is 3 Monit poll cycles. If Monit is setup with a short poll cycle, e.g. 5 seconds or if Monit is verifiable up, but for some reasons slow to send reports it can be useful to increase the *skew* value to avoid getting false alerts because Monit did not report in on time.

Host Groups

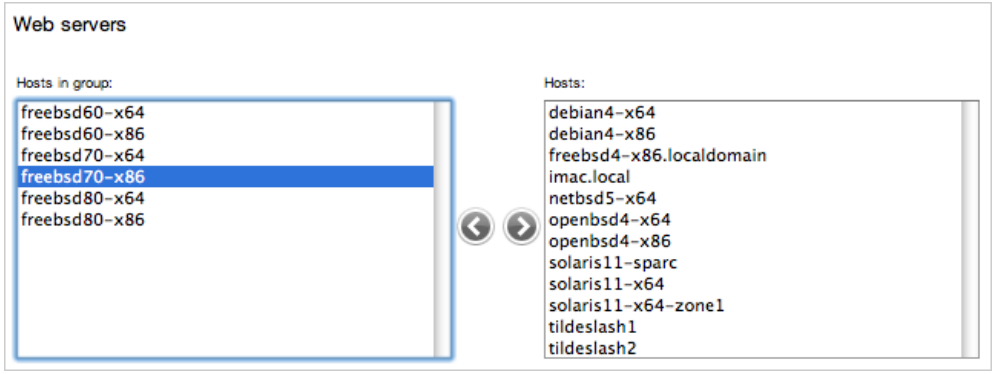
A Host Group is a collection of Hosts that can logically be grouped together, for instance, by functionality or by organizational or geographical location. In the Status and Events page you can drill-down on Host Groups to view data for a set of Hosts. You can also create alert rules that applies to a Host Group.





To create a new host group, select the Add new host group link

Move the mouse cursor over a row in the host group table to display controls for *editing, expanding* or *deleting* a group. To delete a group, click the red icon, to change the group, click the white icon. The blue icon expand the host group and display the group's content;



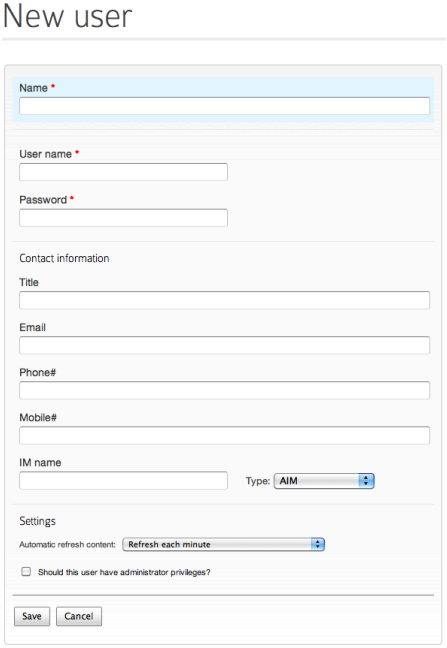
Select hosts from the list to the right and click the grey left arrow to *add* hosts to the group. Select hosts from the left list and click the grey right arrow to *remove* hosts from the group. To select more than one host from the lists, hold down the ALT or ⌘ key when selecting rows.

Users

The Users page list all users allowed to login to M/Monit. M/Monit is installed with two default accounts:

Account	Password	Role
admin	swordfish	Administrator
monit	monit	User

You can change or remove the default accounts, but remember to keep at least one account with administrator privileges to be able to manage M/Monit (see below). To add a new user, click the Add new user link. To edit an existing user, click the user's table row.



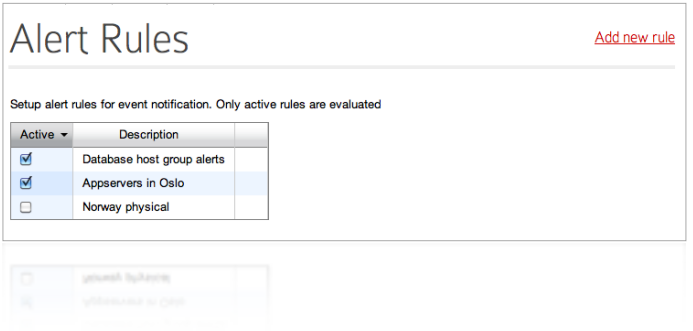
New user. In the Name field, enter the new user's full name. A user name (uname) is required to be able to login to M/Monit. The user name is immutable and once created may not be changed. Password is stored in the database as a MD5 encrypted text string. The other fields are optional, but contact information can come in handy if more people are using M/Monit. If you add email and GTalk/Jabber information you will be able to select the user from the Alert page to receive alerts.

Settings. *Automatic refresh* specifies if, and how often tables and charts in the application should be automatically refreshed. For instance, selecting 5 seconds will update charts in the dashboard page every 5 seconds, as well as the status table and events log. You can also disable refresh, this means that a table or a chart will require a page reload to update. The default refresh rate is one minute.

Check the *Access Rights* checkbox if this user should have administrator privileges. This grants access to all pages and functions in M/Monit. Users *without* administrator privileges cannot access Admin pages nor can they access a host status page (so they cannot restart or stop services on a host).

To **delete a user**, first selected the user from the table then click the delete button, or you can click this ⊗ icon in the rightmost column of the users table.

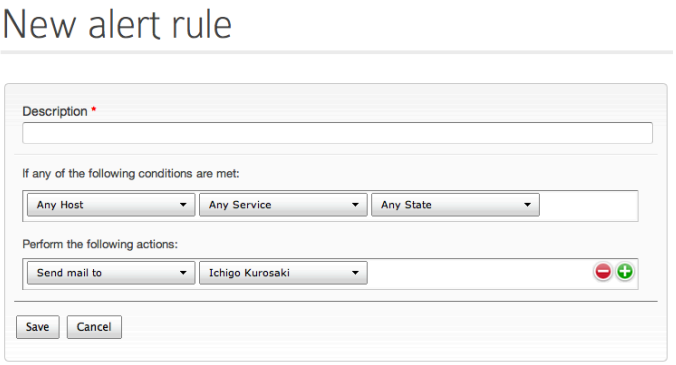
Alert Rules



When an event message is received from Monit, M/Monit will check the event against a list of user defined rules and if a match is found, the actions defined for the Rule are executed.

This page list all rules defined in M/Monit. In the list, a checkbox is used to indicate if a Rule is active or not. M/Monit will only test incoming events against active rules, non-active rules are ignored.

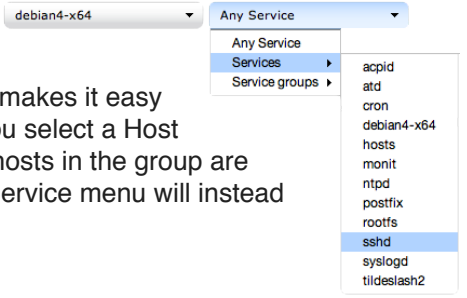
To add a new rule, click the Add new rule link. To edit an existing rule click the rule row in the table.



A Rule is specified as an *if-then* statement. That is, IF a set of conditions are meet THEN perform one or more actions. A rule evaluate to *true* if any condition match the incoming event. A **condition** is a tuple of a Host/ Group, a Service/Group and an event State. This can be read as, if the event originated from a certain host and for a certain service and with a specific event state, then the condition match.

You can specify as many condition rows as you want. A new row is created by clicking the green plus icon and an existing row can be deleted by clicking the red minus icon. To get a notification if a service failed and another when the service comes back up again, create one condition row with [host, service, failed] and one row with [host, service, succeeded]. A row of [Any Host, Any Service, Any State] is a "catch all" condition and will evaluate to true for any event. Such a row can obviously generate a lot of alert messages.

Contextualized drop-down menus

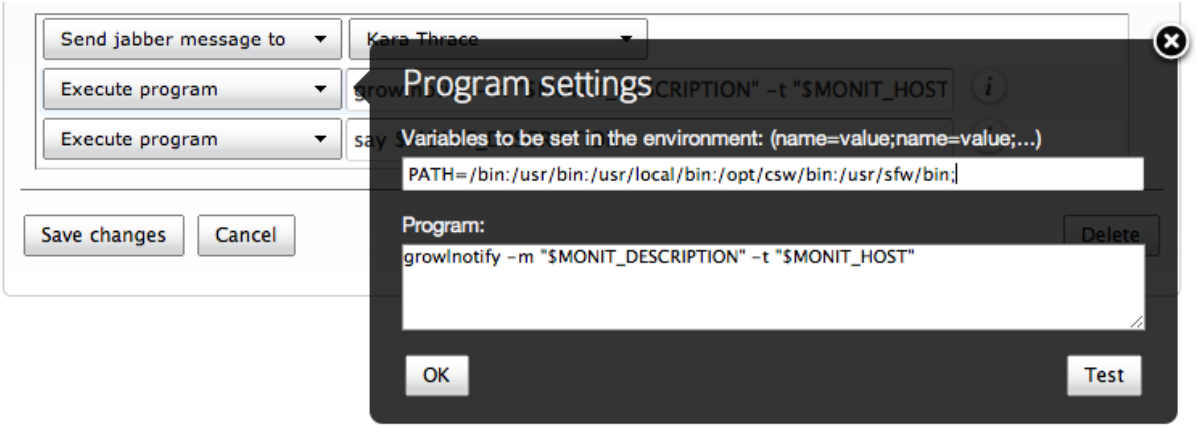


If you select a Host first, the Service submenu will only list Services for the selected Host. This makes it easy to setup an alert for a specific service on a host. The same is the case for Service Groups. If you select a Host Group in the first drop-down menu, only Services and Service Groups that are common for all hosts in the group are listed in the Service menu. If you do not select a host but keep the “Any Host” value, then the Service menu will instead list all services registered in M/Monit.

Alert actions

What M/Monit should do when a rule evaluates to true is specified in one or more **actions**. M/Monit currently supports sending an email, sending a Jabber/GTalk message and executing a program or a script as an action.

- **Email and Jabber/GTalk**. The servers M/Monit should use for sending email and jabber messages are specified in the admin page. There are two ways to specify email. Either by selecting a user registered in M/Monit with an email or by specifying an address directly. The [Send mail to] drop-down target menu contains all users in M/Monit with an email address. If the value is [nobody] then it means that M/Monit cannot find any users with an email address. Likewise for Jabber/GTalk; Users must have a Jabber or GTalk account specified to be listed in the user target menu.
- **Program**. The program M/Monit should execute is specified in the input field. The program is executed by M/Monit using */bin/sh*. You can therefor write a shell script *directly* into the input field or you can simply call an external program. If you want to test the program first or if you need to set specific environment variables for the program, click the white i-icon to the right of the input field. This will pop up a panel where you can add environment variables as well as test execution and check that your program does not return any errors. Once satisfied that the program works as it should, click the OK button.



A set of environment variables is made available to the program at execution time, describing the event that occurred:

- MONIT_EVENT: A short string describing the event that occurred
- MONIT_SERVICE: The name of the service generating the event
- MONIT_DATE: The date and time the event occurred
- MONIT_HOST: The name of the Monit Host the event originated from
- MONIT_ACTION: The name of the action which was performed by Monit
- MONIT_DESCRIPTION: A description of the event condition. I.e. why the event was sent

How to prevent Monit from also sending alerts

If Monit has been setup to send alerts and you want M/Monit to take over the alert responsibility (recommended) then here is a recipe for how to prevent Monit from sending alerts so you don't get double up with alert messages. This will also simplify Monit's configuration file.

1. Remove any "set mailserver", "set mail-format" and "set alert" statements in .monitrc
2. Likewise, remove any standalone alert statements with recipients in *.monitrc*.

For instance if you have:

```
set mailserver...
set mail-format...
set alert...

check file monit with path "/usr/local/bin/monit"
      if changed checksum then alert
      alert someone@some.address
```

Change this to:

```
check file monit with path "/usr/local/bin/monit"
      if changed checksum then alert
```

Server room at night
Monit poll cycle started
alarm quietened

Appendix

server.xml

The configuration file for M/Monit is *server.xml* and in this chapter we describe the configuration directives used in this file.

Directory and file names

If you specify a file that begin with "/" the server will use that absolute path. If the filename does not begin with "/", the value of the M/Monit home directory is prepended. The M/Monit home value is automatically computed based on the location of the mmonit binary. For instance if the mmonit binary is located in "/usr/local/mmonit/bin/mmonit" the mmonit home directory is "/usr/local/mmonit".

<Server>

The Server element represents the entire Container and is the single outermost element in the server.xml configuration file. Only one Service elements may be nested inside a Server element

Attributes	Description
N/A	No attributes are defined for this element

<Service>

A Service element represents the combination of one or more Connector components that share a single Engine component for processing incoming requests. The only components that may be nested inside a Service element are one or more Connector elements, followed by exactly one Engine element.

Attributes	Description
N/A	No attributes are defined for this element

<Connector>

The Connector element represents a Connector component that supports the HTTP/1.1 protocol. It enables the M/Monit Servlet Container to function as a stand-alone web server, in addition to its ability to execute servlets. A particular instance of this component listens for connections on a specific TCP port number on the server. One or more such Connectors can be configured as part of a single Service, each forwarding to the associated Engine to perform request processing and create the response.

At server startup, the Connector will create a pool of servlet request processing threads. The maximum number of threads in the pool is specified by the attribute; *processors*. 10 processor threads are usually more than enough. If you change this, the number of processor threads should probably not exceed 5 x number of CPU cores on the system. Increasing the number of processing threads may or may not increase the throughput and speed of the server, in fact it may decrease the performance since more threads means more overhead and context switching in the kernel.

Each incoming servlet request requires a thread for processing. Usually, only CPU bound operations are performed in the servlet thread, while a separate Container thread handle il/o bound operations for all servlet requests and responses. If more simultaneous servlet requests are received than can be handled by the currently available thread pool, requests are queued up inside the Connector, up to the systems maximum available descriptors and when a processor thread becomes available it will immediately start to consume requests from the queue. Operating Systems allows normally anywhere from 128 to 1024 simultaneously open descriptors per process. It is recommended to increase the limits of open file descriptors available to a process before mmonit is started from the console. Use e.g. `ulimit -n 1000` in the console before starting mmonit.

The attribute, *processorTimeout* sets the number of seconds a processor thread will wait for more work before timeout. The Connector increase and reduce the number of processor threads available depending on the work load. The default timeout value is 30 seconds.

The attribute, *connectionTimeout* specify the number of seconds a Connector will wait, after accepting a connection, for the client to send a HTTP request. The default value is 30 seconds.

The attributes, *address* and *port*, specify respectively the network interface M/Monit binds to and the port number M/Monit listen to for incoming connections. Address may be specified as an IP address string, as a host name or you can use '*' to bind to all available interfaces.

The attribute *ipversion* specify the IP-version the Connector should use. If not specified, IP-version 4 will be used. To support both IPv4 and IPv6, specify two Connectors, one setting ipversion to 4 and the other setting ipversion to 6.

By default, a non-SSL HTTP/1.1 Connector is established on port 8080. You may also enable a SSL HTTP/1.1 Connector on port 8443 by uncommenting the second Connector entry in server.xml. To make a Connector secure and use SSL, set the secure attribute to true - that's all. The Container will use either SSLv3 or TLSv1. SSLv2 is never used. In addition, each virtual Host must specify a

certificate file to be used for that specific Host. It is possible to run the Container with both a secure Connector using SSL and a non-secure Connector.

By default, DNS lookup is disabled and the Access Logger will log the IP address instead of the host name. You can enable DNS lookup by setting the enableLookups attribute to "true", but notice that DNS lookups will have an adverse impact on performance if you use an Access Logger.

Attributes	Description	Type	Example
enableLookups	Set to true if you want the AccessLogger to log the Host name from incoming clients connections. Set to false to skip the DNS lookup and return the IP address in String form instead (thereby improving performance). By default, DNS lookups are disabled.	Boolean	enableLookups="true"
redirectPort	If this Connector is supporting non-SSL requests, and a request is received for which a matching <security-constraint> requires SSL transport, the Container will automatically redirect the request to the SSL Connector port number specified here.	Number	redirectPort="8443"
secure	Define if the Connector should use SSL for incoming client connection. If true implies the https scheme otherwise the http scheme. If selected you must also define the certificate file in the Host element and the IP-address of the Host.	Boolean	secure="false"
address	The address the Connector will accept connect requests to. If the address is not defined the Connector will default to accepting connections on any/all local addresses	String	address="*" or address="64.87.72.95"
connectionTimeout	The number of seconds a Connector will wait, after accepting a connection, for client to send a HTTP request. The default value is 30 seconds.	Number	connectionTimeout="30"
processors	The maximum number of request processing threads that will be used to serve HTTP requests. The Connector create threads as needed and up to the configured number of processors. The default value is 128 processor threads.	Number	processors="15"

Attributes	Description	Type	Example
processorTimout	Sets the number of seconds a processor thread will wait for more work before timeout. The Connector increase and reduce the number of processor threads available depending on the work load. Default value is 30 seconds.	Number	processorTimout="60"
port	The TCP port number on which a Connector will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. The default port is 8080.	Number	port="8080"
ipversion	Specify the IP-version to be used by the Connector. Legal values are either 4 or 6. Unless specified, IP-version 4 will be used.	Number	ipversion="6"

<Engine>

The Engine element represents the entire request processing machinery associated with a particular Service. It receives and processes all requests from one or more Connectors, and returns the completed response to the Connector for ultimate transmission back to the client. Exactly one Engine element MUST be nested inside a Service element.

The Host defined in the *defaultHost* attribute is used to process Requests directed at Virtual Hosts not configured in this configuration file. The default Host will also handle HTTP/1.0 based requests without a Host header.

The *filecache* attribute is used to set the file cache size for the Engine. The Engine cache static files in memory, to speed up transmission. This is particular useful for secure SSL transmissions. The cache size attribute is set in megabytes (MB) and the default size is 10MB. If the size is set to a value less than 1MB it is ignored and the maximum cache size set to 1MB. You can disabled the cache by setting filecache to 0. Disabling the file cache is strongly advised against for production systems.

You can nest one or more Host elements inside the Engine element, each representing a different virtual host associated with this server. At least one Host is required, and one of the nested Hosts MUST have a name that matches the name specified for the defaultHost attribute, mentioned above.

You can also nest at most one instance of the following utility components inside an Engine element:

- ErrorLogger - Configure an error logger that is used by the Server to dump error and warning messages.
- Realm - Set the security realm database used to authenticate individual users and used by the M/Monit application.

Attributes	Description	Type	Example
name	The Engine name. Merely for documentation purpose.	String	name="StandAlone"
defaultHost	The default host name, which identifies the Host that will process requests directed to host names on this server, but which are not configured in this configuration file. This name MUST match the name attributes of one of the Host elements nested immediately inside.	String	defaultHost="localhost"
filecache	The file cache size for the Engine. Normally, the Engine cache "small" static files in memory to speed up transmission and reserve the sendfile() OS "zero-copy" mechanism for transmitting larger files. In some situations the "zero-copy" mechanism cannot be used, especially when files are transmitted over SSL. In this case larger files are also cached to speed up transmission. The cache size attribute is set in MB and the default size is 10Mb.	Number	filecache="5MB" or to disable filecache="0"

<Host>

The Host element represents a virtual host, which is an association of a network name for a server (such as "www.mycompany.com" with the particular server on which the Servlet Container is running. This name must be registered in the Domain Name Service (DNS) server that manages the Internet domain you belong to - contact your Network Administrator for more information.

In many cases, System Administrators will wish to associate more than one network name (such as www.mycompany.com and company.com) with the same virtual host. This can be accomplished by using the alias attribute. You may add as many Host aliases as you like, but note that a Host alias must also be a valid DNS name.

The appBase attribute defines the application root directory for the Host. This directory may contain web applications to be deployed on this virtual host.

The certificate attribute is used to specify the SSL certificate file for the Host. Certificates must be in the PEM format and the file must contain the following entries in this order: The Host certificate private key, the Host certificate and finally, unless this is a self-signed certificate, the certificate of the authority that signed the Host certificate. When SSL is used, the Host IP-address is needed for the server to know which Host to route the connection to and you must specify the IP-address of the Host by using the address attribute.

If you don't already have a SSL certificate you can create a self-signed certificate yourself using this OpenSSL command:

```
openssl req -new -newkey rsa:2048 -x509 -days 730 \
        -nodes -out mmonit.pem -keyout mmonit.pem
```

You can nest one or more Context elements inside the Host element, each representing a different web application associated with the virtual host.

You can also nest at most one instance of the following utility components by nesting a corresponding element inside your Host element:

- *AccessLogger* When you run a web server, one of the output files normally generated is an access log, which generates one line of information for each request processed by the server, in a standard format.
- *Logger* A Logger shared by all Contexts related to this virtual host. The Logger will process all log messages for a Host, plus messages from Contexts and Servlets associated with the Host.

Attributes	Description	Type	Example
appBase	The Application Base directory for this virtual host. This is the pathname of a directory that may contain web applications to be deployed on this virtual host. You may specify an absolute pathname for this directory, or a pathname that is relative (to the mmonit home directory).	String	appBase="webapps"
name	The network name of this virtual host, as registered in your Domain Name Service server. One of the Hosts nested within an Engine MUST have a name that matches the defaultHost setting for that Engine. See the Host Name Alias attribute below for information on how to assign more than one network name to the same virtual host.	String	name="localhost"
alias	An alias for the host. That is; a DNS C record. You can add more than one alias attribute, as long as the Host alias represents a real DNS record.	String	alias="www.foobar.com" alias="foobar.com"

Attributes	Description	Type	Example
certificate	Specify the SSL certificate file for the Host. Entries must be in the PEM format and must contain the following entries in this order: The Host certificate private key, the Host certificate and finally, the certificate of the authority that signed the Host certificate.	String	certificate="conf/mmonit.pem" or with an absolute path certificate="/etc/ssl/certs/myhost.pem"
address	When SSL is used, the Host IP-address is needed for the Container to know which Host to route the connection to.	String	address="62.48.16.37"

<Context>

The Context element represents a web application, which is run within a particular virtual host. A web application is a collection of servlets, html documents, images and other resources put in a directory structure with a standard layout.

The web application used to process each HTTP request is selected by the Container based on matching the longest possible prefix of the Request URI against the context path of each defined Context. Once selected, that Context will select an appropriate servlet to process the incoming request, according to the servlet mappings defined in the web application deployment descriptor file (which MUST be located at /WEB-INF/web.xml within the web app's directory hierarchy).

You may define as many Context elements as you wish, nested within a Host element. Each such Context MUST have a unique context path, which is defined by the path attribute. In addition, you can define a Context with a context path equal to a zero-length string. This Context becomes the default web application for this virtual host, and is used to process all requests that do not match any other Context's context path. If such a context is not defined in this configuration file, the Servlet Container will create a default context with ROOT as the docBase directory.

Each Context utilize a Session Manager to manage HTTP sessions for the Context. The maxActiveSessions attribute specify the maximum number of sessions that will be created for the Context. The manager expire idle sessions after *sessionTimeout* seconds. It is possible to turn off sessions for a Context by setting the attribute maxActiveSessions to 0 or to a negative value. Note that sessions are not enabled unless maxActiveSessions is defined and set to a positive value.

Attributes	Description	Type	Example
docBase	The Document Base (also known as the Context Root) directory for this web application, or the pathname to the web application archive file (if this web application is being executed directly from the WAR file). You may specify an absolute pathname for this directory or a WAR file, or a pathname that is relative to the appBase directory of the owning Host.	String	docBase="examples"
path	The context path of this web application, which is matched against the beginning of each request URI to select the appropriate web application for processing. All of the context paths within a particular Host must be unique. If you specify a context path of an empty string (""), you are defining the default web application for this Host, which will process all requests not assigned to other Contexts.	String	path="/examples" or for the default Context, path=""
reloadable	Set to true if you want the Servlet Container to monitor classes in WEB-INF/classes/ for changes, and automatically reload the web application if a change is detected. This feature is very useful during application development, but it requires runtime overhead and is not recommended for use on deployed production applications.	Boolean	reloadable="true"
sessionTimeout	The session timeout in seconds for this web application. The default value is 900 seconds (15 minutes)	Number	sessionTimeout="1800" (30 minutes)
saveSessions	Set to true if you want to enable persistent Sessions. I.e. Sessions are stored in the Realm database and retained during application restart. The default value is false.	Boolean	saveSessions="true"
maxActiveSessions	The maximum number of active sessions that will be created by this Context. The default is 4096. If this attribute is not used or the value is 0 or less the Context will not support sessions.	Number	maxActiveSessions="2000" or for no sessions, maxActiveSessions="0"

<Realm>

The Realm element specify the database to be used by M/Monit and the underlying security realm to authenticate individual users and store HTTP sessions. If the Realm database is not defined, M/Monit will not start and abort its operation.

Connections from M/Monit to the Realm database is maintained by a database Connection Pool. The url attribute specify the connection to the database server on a standard URL format. The format of the connection URL is defined as:

```
database://[user:password@][host][:port]/database[?[property1=value1][&property2=value2]...]
```

The property names, user and password are always recognized and specify how to login to the database. Other properties depends on the database server in question. User name and password can alternatively be specified in the authentication part of the URL. If port number is omitted, the default port number for the database server is used.

The optional attributes, *minConnections* and *maxConnections*, specify respectively, the minimum number of concurrent available connection and the maximum number of database connections that can be created. The pool will dynamically increase and reduce the number of active Connections in the pool between minConnections and maxConnections depending on the load. If not set, minConnections is 5 and maxConnections is 20.

The attribute *reapConnections* specify if the Connection Pool should run a reaper thread, which will close and remove unused Connections from the Pool. The value is sweep time in seconds. I.e. the reaper thread will sleep for reapConnections seconds,wake up, clean up the pool if necessary and go back to sleep. If this attribute is not set, the Connection Pool will not start with a reaper thread. It is highly recommended to activate a reaper thread for the connection pool so stale connections and excess connections automatically are closed down.

The location of the default SQLite Realm database may be given as an absolute or relative path. If relative, the absolute path is computed relative to M/Monit home. As mentioned above, instead of SQLite you may use MySQL or PostgreSQL as demonstrated in the commented out Realms in server.xml.

A realm operates according to the following rules:

- When a user attempts to access a protected resource for the first time, M/Monit will call the authenticate() method of this Realm. Thus, any changes you have made to the database directly (new users, changed passwords or roles, etc.) will be immediately reflected.
- Once a user has been authenticated, the user (and his or her associated roles) are cached within the server for the duration of the user's login. For FORM-based authentication, that means until the session times out or is invalidated; for BASIC authentication, that means until the user closes their browser.
- The information in the user database is controlled by the M/Monit admin/users/ page.

Only one Realm element should be defined per Server and the element is defined within an Engine element.

Attributes	Description	Type	Example
URL	The database to be used by M/Monit and the underlying security realm to authenticate individual users and store HTTP sessions. Connections to the database is maintained by a Connection Pool.	URL	url="sqlite:///db/mmonit.db"
minConnections	The minimum and initial number of concurrent available connections to the Realm database. The default value is 5 connections.	Number	minConnections="10"
maxConnections	The maximum number of connections to the Realm database.	Number	maxConnections="50"
reapConnections	Start the Realm Database Connection Pool with a reaper thread which will close and remove unused Connections from the Pool. Value in seconds, inside the range [1..86400] sec	Number	reapConnections="300"

<ErrorLogger>

An ErrorLogger is used for logging debug and error messages (including stack tracebacks). The ErrorLogger does not buffer data but writes directly to the log file. Each entry in the log file is prefixed with a timestamp. The ErrorLogger component is optionally and if it is not defined the Server will write error messages to stderr.

Only one ErrorLogger element should be defined per Server and the element is defined within an Engine element.

Attributes	Description	Type	Example
directory	Absolute or relative pathname of a directory in which log files created by this Logger will be placed. If a relative path is specified, it is interpreted as relative to M/Monit home.	String	directory="/usr/local/mmonit/logs"
fileName	The name of the log file the ErrorLogger will write to. The filename must not be prefixed with a path.	String	fileName="error.log"

<AccessLogger>

The AccessLogger create log files in the same format as those created by standard web servers. These logs can later be analyzed by log analysis tools to track page hit counts, user session activity, and so on. An AccessLogger is associated with a virtual host and will record ALL requests processed by that Host.

The AccessLogger flush to the log file every 10 seconds or writes every 32Kb of log entries if this is reached sooner.

The log format used by an AccessLogger is the Common Log Format plus entries for Referer and User-Agent. An entry in the log file may look like:

```
64.87.72.95 - admin [18/Oct/2009:01:10:21 +0200] "GET /status/ HTTP/1.1" 200 3707 "http://localhost/" -
```

The first field (64.87.72.95) is the hostname or IP address of the connecting machine. The second is a username from an ident lookup. If no ident lookup was performed the '-' character is used. The third is the auth-username if authentication was performed. The fourth is the timestamp for the request. The fifth the HTTP request sent to the server. The sixth field is the HTTP status code returned in the response. The seventh field is the response-size, that is, the size of the response entity, not including HTTP headers. If no entity was returned in the response the value is zero. The last two fields contains HTTP headers sent by the Browser. The field are respectively the HTTP Referer and the User-Agent. Note that the Browser may opt not to send any these HTTP headers in which case the last two fields will have the value "-".

Attributes	Description	Type	Example
directory	Absolute or relative pathname of a directory in which log files created by this Logger will be placed. If a relative path is specified, it is interpreted as relative to M/Monit home.	String	directory = "/usr/local/mmonit/logs"
fileName	The name of the log file the AccessLogger will write to. The filename must NOT be prefixed with a path.	String	fileName = "localhost_access.log"

Attributes	Description	Type	Example
rotate	The rotate attribute may be used to rotate the log file (without having to restart the mmonit server). The value is either "day", "week" or "month". If the value is "day" then on the first logged message after midnight each day, the current log file will be closed and renamed with a postfix date and a new log file is opened with the file name given in the fileName attribute. If the value is "week" then on the first logged message after midnight each Saturday the log file is rotated. Likewise if the value is "month" then on the first logged message in a new month the log file is rotated. The current log file, that is, the file the server writes to is always the file given in the fileName attribute.	String	rotate="month"
pattern	This attribute is reserved for future use and will be used to select a particular log format.	String	pattern = "common"

<Logger>

A Logger is associated with a Host, and Servlets registered within the Host Context can use a Logger for writing log messages to a log file. M/Monit uses this log file to write application specific data.

Attributes	Description	Type	Example
directory	Absolute or relative pathname of a directory in which log files created by this Logger will be placed. If a relative path is specified, it is interpreted as relative to mmonit home.	String	directory="/usr/local/mmonit/logs"
fileName	The name of the log file the Logger will write to. The filename must NOT be prefixed with a path.	String	fileName="localhost.log"
timestamp	Set to true to cause all logged messages to be prefixed with a timestamp (the default). Set to false to skip stamping.	Boolean	timestamp="true"

<License>

M/Monit is a licensed product and M/Monit comes with an evaluation license which will expire. If you have purchased a full license, replace the License element in server.xml with your new key and *restart* M/Monit.

Attributes	Description	Type	Example
owner	The license owner. The owner is associated with the key and cannot be changed.	String	owner=”Tildeslash”

M/Monit behind proxy

M/Monit can be used from behind a proxy server. Here is an example on how to configure Apache Proxy in front of M/Monit. In this example M/Monit listens on http://192.168.1.10:8080 and we configure Apache so M/Monit is accessible via this Apache Proxy URL, http://<apache's address>/mmonit/

```
ProxyPass /mmonit/ http://192.168.1.10:8080/
ProxyPassReverse /mmonit/ http://192.168.1.10:8080/
<Location /mmonit/>
     Order deny,allow
     Allow from all
</Location>
```