Piotr Kodzis

AKK@DA Users' Guide

version 1.0

AKK@DA Users' Guide

version 1.0

e-mail: piotr.kodzis@yahoo.pl
home page: http://akkada.eu/
project page: http://sourceforge.net/projects/akkada/

Table of Contents

Chapter 1

1. Overview of AKK@DA

1.1. Preface

This manual describes AKK@DA version 0.75. It covers installation, running, configuration, day-to-day maintenance tasks and using the web based GUI. If you have any comments or suggestions, please e-mail piotr.kodzis@yahoo.pl.

1.2. Overview

AKK@DA is a network monitoring system designed for small and mid-sized computer networks. Its purpose is to quickly detect system or network fault and to display information about any detected problems for the administrator. AKK@DA is designed as a pro-active network monitor. It does not need to be provided with information from any agents, systems, etc. It collects information automatically every single minute (this period can be shortened even to 1 second if needed). Almost all services of the monitored hosts are discovered automatically.

1.3. Features

AKK@DA allows you to perform the following tasks:

- monitoring host availability, resources and network services (probes use SNMP, ICMP, raw TCP, SSL, DNS, UCDavis script extensions and many others for checking host health and collecting data),
- automatic discovering of services available on a/the host like CPU, RAM, network interfaces, disks, processes, etc, etc (it's not necessary to configure it manually, you simply have to add a new host (via the web based GUI) and AKK@DA will discover this host's services for you),
- collecting performance data about monitored services (fully RRDTool integrated),
- service flap detection,
- adding your own probes,
- adding support to SNMP devices not supported by AKK@DA through the template based model,
- multi SNMP agents host support
- easily using the web based GUI with support features such as:
    - alarm presentation with the option to correlate and approve alarms,
    - filterable logs,
    - RRDTool based performance graphs,
    - configuration of monitored services or hosts,
    - AKK@DA system status and management,
    - support of contact groups,
    - user and group management,
    - mail notifications, GTalk notifications,
    - right management (you can manage rights of the group to every single service),

- tree-based host organization - allows organizing monitored resources in easy to browse and quick to navigate schema,
- view support (you can arrange monitored services in groups; search results might be saved as views),
- dashboard,
- displaying information in compact mode allows you to estimate the host health at the first sight,
- constantly available context menus allow quick access to necessary options and pages,
- some complex reports such as time synchronization in the monitored network, Cisco inventory report, etc.,
- web interface looks like standard MS Windows; all GUI written in DHTML & JavaScript - no applets, ActiveX controls or other strange modules.

1.4. System architecture

AKK@DA is a group of autonomous programs which co-operate in the process of network monitoring. All these programs are daemons managed by the main AKK@DA daemon akkada.pl. There are two groups of daemons:

- core AKK@DA programs which are responsible for data processing,
- probes which provide discovery logic, test monitored hosts and provide a presentation layer for the GUI.

1.4.1. Core programs

Core programs are named **nm-*.pl**. Most of them have to be activated to make AKK@DA work. Only one instance of each of them can exist. Below is a brief description of functions of these modules:

nm-actions_broker.pl	manages action requests submitted by probes (e.g. used by the notification mechanism),

mm-actions_executor.pl	executes actions reported by nm-actions_broker.pl,
nm-available.pl	constantly tests availability of all monitored hosts using ICMP protocol,
nm-available2.pl	next generation of nm-available.pl which uses the model of monitored network structure for detecting complex network faults,
nm-db_watch.pl	informs all other modules whenever an entity is added or deleted from the configuration,
nm-discover.pl	periodically discovers supported services on monitored hosts using logic provided by probes,
nm-icmp_monitor.pl	monitors all configured host ICMP delays, packet losses, jitter and collects long term statistics,
nm-job_planner.pl	organizes probe tests,
nm-status_calc.pl	calculates host status based on the status of their services,
nm-sysstat.pl	collects AKK@DA internal performance data,
nm-tree_cache.pl	keeps an up to date internal AKK@DA cache used to share information between internal programs and the web based GUI.

Table 1. Modules list

For more information, see chapter **The core of AKK@DA**.

1.4.2. Probes

Probes are named np-*.pl. They test monitored hosts. There are many probes because each type of service needs to have a separate probe (except for the np-snmp_generic.pl probe which is based on a templates model and is able to monitor different types of services). Many instances of the specific probe can be activated simultaneously because a single probe instance can test only a limited number of services.

1.5. Hardware requirements

Fast disks, good CPU and minimum 1 GB of RAM are the minimal requirements. AKK@DA checks all services every minute. It is a lot to do, so consequently its processes consume a lot of system resources. Choosing proper hardware depends on the service count which AKK@DA will have to monitor. It is strongly recommended to have a separate server for the AKK@DA system.

For example:

- PC, Pentium III 800 MHz, 512 MB RAM with IDE disks is able to monitor around 700 services,
- Server HP DL 380, 2 x Pentium 4 2.9 GHz, 2 GB RAM with SCSII disks is able to monitor around 7000 services.

Discovery process
In the case of MS Windows the server automatic discovery process discovers around 80 services, in the case of Unix server ~20 services; in the case of the network device count the number of discovered services depends on the number of interfaces in the box. There is an option to stop monitoring unnecessary services in order to improve performance, if needed.

1.6. Operating system requirements

AKK@DA was developed on Linux OS (Fedora). It was also tested on RedHat Enterprise Linux and Gentoo Linux. Additionally the following software has to be installed:

- **MySQL**, version 4.x or later (http://www.mysql.com/),
- **Apache**, version 2.x with **mod_perl** version 2 (http://perl.apache.org/),
- **fping** (http://www.fping.com),
- **nmap** (http://insecure.org/nmap/),
- **RRDtool**, version 1.2 or later (http://oss.oetiker.ch/rrdtool/)
  ☝ compiled with option **--enable-perl-site-install** (see **./configure --help** from RRDtool sources),

- **perl**, version. 5.8 or later (http://www.perl.org/)
  ☝ compiled with thread support,

- **Graphviz** (http://www.graphviz.org)
  ☝ only if nm-available2.pl will be used,

- the following **perl modules** (http://search.cpan.org/):

  Authen::SASL
  Bit::Vector
  Cache::File
  CGI
  CGI::Compress::Gzip
  CGI::Session
  Crypt::DES
  Date::Manip
  DBI
  Digest::HMAC
  Digest::MD5
  Digest::SHA1
  Error
  File::NFSLock
  Graph
  Graph::Easy
  HTML::Parser
  HTML::Table
  HTTP::Request::Common
  IO::Socket::INET
  IO::Socket::SSL
  IPC::Open3
  LWP::Parallel::UserAgent
  Mail::Sender
  Math::RPN
  NetAddr::IP
  Net::DNS
  Net::Gadu
  Net::IP
  Net::SNMP
  Net::SSH::Perl
  Net::Telnet::Cisco
  Net::XMPP
  Number::Format
  Pod::Escapes
  Pod::Simple
  Proc::ProcessTable
  Test::Builder::Tester
  Test::More
  Test::Pod
  Time::Date
  Time::HiRes
  Time::Period
  XML::Simple
  XML::Stream.

Date and time
Your OS date and time should be correct otherwise AKK@DA can have problems updating RRD database files. It is recommended to use the NTP daemon to keep your OS time up to date. See http://www.ntp.org/ for more information about installing and using the NTP daemon.

1.7. Client requirements

Web based graphical user interface is written in HTML, DHTML and JavaScript. It does not use any ActiveX controls, Java applets or other type of client-side software. It supports the following web browsers:

- **Microsoft Internet Explorer** 6 or later
- **Mozilla Firefox** 1.5 or later

The web based GUI also works fine with **Opera**, but in this case its context menus don't work.

Chapter 2

2. Installing AKK@DA

2.1. Operating system preparation

General information

The dedicated server should be prepared for AKK@DA because AKK@DA uses a lot of the server resources. It is recommended that this server has an access to the Internet for easier installation of necessary software and for the purpose of NTP time synchronization.

Operating system should be installed including developer tools (libraries, C++, compiler, etc.). There are no special needs for the partitioning of disk space. AKK@DA needs some GB of disk space to keep RRD database files, but the amount of this space greatly depends on the number of monitored services (approximately 1MB of disk space per service is needed). If your OS allows, it is recommended to have 64MB of ramdisk space (in the case of RedHat and Fedora ramdisk is available by default at the **/dev/shm/** directory). All software listed in the chapter *"Operating system requirements"* has to be installed. For more details and installation procedures, refer to their manuals.

To install AKK@DA you have to switch to user root.

Perl modules

Installing perl modules might be a long and laborious process. To make it easier it is suggested to use CPAN mechanism, which is possible only if the server has access to the Internet. The first time you run installation through CPAN the installation program will ask you a few configuration questions. An example installation process of the Date::Time module:

```
[root@localhost]# perl -MCPAN -e "install Date::Time"
```

For more information, refer to CPAN documentation at http://www.cpan.org/.

System users, groups and environment

AKK@DA should be installed in directory **/akkada**.

It is possible to install AKK@DA in a different directory but in such case the installation procedure and AKK@DA initial configuration should be modified and adjusted manually. However, this is complicated so it is recommended not to change this directory.

User **akkada** and group **akkada** should be created as well (home directory for **akkada** user has to be set to **/akkada**):

```
[root@localhost]# groupadd akkada
[root@localhost]# useradd –g akkada –d /akkada akkada
```

The **akkada** user has to be added to the main group of the user which is used by the Apache server (see the **/etc/groups** file and the Apache **httpd.conf** file).

Environment variable **AKKADA** has to be set for all users – e.g. by adding to the **/etc/profile** file the following lines:

```
export AKKADA=/akkada
export KEEPENV="PATH AKKADA"
```

The host name of the server has to be resolvable by the local resolver – it's not importand whether it is based on the local **/etc/hosts** file or DNS servers.

fping and nmap

fping and **nmap** binaries should have SUID bit set to 'on' to make them usable for **akkada** user.

```
[root@localhost]# chmod +s `which fping`
[root@localhost]# chmod +s `which nmap`
```

2.2. Obtaining AKK@DA

AKK@DA is available on the Internet at http://akkada.tivi.net.pl/download/. It is important to get the latest version of AKK@DA which is called **akkada-current.tar.gz**.

2.3. Installing

Make sure the MySQL server is up and running before you start installing AKK@DA.

1. Uncompress **akkada-current.tar.gz**:

```
[root@localhost]# cd /
```

```
[root@localhost]# gunzip < akkada-current.tar.gz | tar xvf –
```

2. Link Perl binary:

```
[root@localhost]# cd /akkada/bin
[root@localhost]# rm -f perl
[root@localhost]# ln -s `which perl` perl
```

3. Initialize database:

```
[root@localhost]# cd /akkada/bin
[root@localhost]# mysql < akkada_db_create.sql
[root@localhost]# mysql akkada < akkada_db_init.sql
[root@localhost]# mysql akkada < akkada_db_users_init.sql
```

4. Update file **/akkada/etc/akkada.shell**:

Name	Default value	Description
OSLogin	akkada	User to run AKK@DA as
OSGroup	akkada	Group to run AKK@DA as
ApacheLogin	apache	User to run Apache as (see Apache configuration file httpd.conf)
ApacheGroup	apache	Group to run Apache as (see Apache configuration file httpd.conf)
MYSQL	/usr/bin/mysql	Full path to MySQL command-line tool
MYSQLDUMP	/usr/bin/mysqldump	Full path to MySQL database backup program

Table 2. /akkada/etc/akkada.shell options

5. Set proper rights for AKK@DA files

```
[root@localhost]# cd /akkada/bin
[root@localhost]#./post_install.sh
```

2.4. Post install configuration of AKK@DA

AKK@DA configuration is kept in many text files in the /akkada/etc directory which contain regular Perl data structures. These data structures are very s trict and there is no place for any latitude. If you are not familiar with Perl data structures please change them very carefully, paying special attention to " ' [ ] ( ) { } ; and comma characters. Always create backup of your configuration files

before changing them. After changing always verify your changes as described in chapter 3 before restarting AKK@DA.

Paths

The paths to mysqladmin, fping, nmap and ntpq binary files should be updated in AKK@DA configuration files to point at binaries in your specific installation. Configuration files are kept in the **/akkada/etc** directory. To find the files in which paths to those tools are located use the **cfgfindraw.sh** tool.

fping example

Here is how to check if the fping path is correct:

```
[root@localhost]# cd /akkada/bin

#check where is your fping located:
[root@localhost]# which fping
/usr/local/sbin/fping

#check how is your AKK@DA configured:
[root@localhost]# ./cfgfindraw.sh fping
/akkada/etc/conf.d/ICMPMonitor.conf:    'fping' => [
/akkada/etc/conf.d/ICMPMonitor.conf:      '/usr/sbin/fping',
/akkada/etc/conf.d/Available.conf:    'fping' => [
/akkada/etc/conf.d/Available.conf:      '/usr/sbin/fping',
```

The example above shows fping is located in /usr/local/sbin/fping but fping pointer exists in two AKK@DA configuration files:

/akkada/etc/conf.d/ICMPMonitor.conf
/akkada/etc/conf.d/Available.conf

and both pointers are incorrect (point at /usr/sbin/fping instead of /usr/local/sbin/fping) and should be corrected. A similar procedure should be repeated for other programs mentioned above.

Database connectivity

By default AKK@DA uses the user **root** with no password to connect to the database, but it is possible to change that if needed. Database connectivity options are kept in the **/akkada/etc/conf.d/Database.conf** file:

```
{
    'Password' => '',
    'DSN' => 'DBI:mysql:database=akkada;host=localhost;port=3306',
    'Username' => 'root',
    'CharSet' => 'latin2',
},
```

Username and **Password** fields define the user used by AKK@DA for connecting to the database and should be modified if needed. The **DSN** field usually shouldn't be modified (for more information about **DSN** meaning, refer to the Perl **DB.pm** module manual). Field **CharSet** shouldn't be modified.

To test the configuration of the database connectivity use command line tool **/akkada/bin/db_conn_check.pl** (for more information, see chapter **Scripts**, subchapter **Command line tools**).

RAM disk

For performance reasons it is recommended to move directories pointed by following options: **FlagsControlDir**, **DataDir** and **TreeCacheDir** (use the cfgfindraw.sh tool to find specific configuration files) to ramdisk (if available). If you decide to use ramdisk it is important to recreate directories on ramdisk every time the server is restarted. The best way to do this is creating the **/akkada/bin/akkada_pre** shell script. AKK@DA start script **/akkada/bin/akkada** always tries to run the **/akkada/bin/akkada_pre** script before starting procedure. An example **akkada_pre** script which checks if directories on ramdisk exist:

```
#!/bin/sh

[ -d /dev/shm/data ] || mkdir /dev/shm/data
chown akkada.apache /dev/shm/data
chmod 770 /dev/shm/data

[ -d /dev/shm/tree_cache ] || mkdir /dev/shm/tree_cache
chown akkada.apache /dev/shm/tree_cache
chmod 770 /dev/shm/tree_cache

[ -d /dev/shm/control ] || mkdir /dev/shm/control
chown akkada.apache /dev/shm/control
chmod 770 /dev/shm/control
```

Number of probes

Edit the **/akkada/etc/conf.d/System.conf** file to disable unneeded probes and adjust probe counts to your needs. Only section **Probes** should be modified (in black below), please don't change the **Modules** section (in red below).

An example **/akkada/etc/conf.d/System.conf** file:

```
{
    'Probes' => {
      'ucd_ext' => 1,
      'nic' => 1,
      'host_resources_process' => 0,
      'host_resources_system' => 1,
      'dns_server' => 1,
      'cisco_css_content' => 0,
      'softax_ima' => 0,
      'softax_ping' => 0,
      'node' => 1,
      'ucd_process' => 1,
      'hdd' => 1,
      'cisco_css_service' => 0,
      'ssl_generic' => 1,
      'cpu' => 1,
      'windows_service' => 0,
      'route' => 0,
      'cisco_css_owner' => 0,
      'cisco_dial_peer_voice' => 1,
      'dns_query' => 0,
      'tcp_generic' => 1,
      'snmp_generic' => 0,
      'icmp_monitor' => 0,
      'bgp_peer' => 0,
      'tcpip' => 1,
      'ram' => 1,
      'ntp' => 1,
    },
```

```
    'Modules' => {
      'tree_cache' => 1,
      'status_calc' => 1,
      'db_watch' => 1,
      'discover' => 1,
      'job_planner' => 1,
      'available' => 1,
      'icmp_monitor' => 1,
    }
}
```

This file defines the number of probes which will be started. 0 means specific probe is disabled. Any future changes in this file need AKK@DA restart.

Verify your configuration as described in chapter 3.

2.5. The Web GUI configuration

The web based GUI uses the Apache server with mod_perl 2. It is suggested to run a separate instance of the Apache server for AKK@DA web GUI purpose. The following configuration should be added to the Apache server configuration (e.g. to the httpd.conf file):

```
DocumentRoot "/akkada/htdocs"
<Directory "/akkada/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

LoadModule expires_module modules/mod_expires.so
<IfModule mod_expires.c>
ExpiresByType text/html A1
</IfModule>

LoadModule perl_module modules/mod_perl.so
PerlRequire "/akkada/bin/startup.pl"
PerlPassEnv AKKADA
PerlPassEnv REMOTE_ADDR

PerlModule Runtime
<Location /gui>
SetHandler perl-script
PerlResponseHandler Runtime
</Location>

PerlModule MyChart
<Location /graph>
SetHandler perl-script
PerlResponseHandler MyChart
</Location>

PerlModule ContactsRuntime
<Location /contacts>
SetHandler perl-script
PerlResponseHandler ContactsRuntime
</Location>
```

```
PerlModule CommentsRuntime
<Location /comments>
SetHandler perl-script
PerlResponseHandler CommentsRuntime
</Location>

PerlModule GraphSysStat
<Location /graphsysstat>
SetHandler perl-script
PerlResponseHandler GraphSysStat
</Location>

PerlModule ImgsRuntime
<Location /imgs>
SetHandler perl-script
PerlResponseHandler ImgsRuntime
</Location>
```

Verify your configuration as described in chapter 3.

2.6. Common problems

Cannot start Apache server

Check Apache server error log:

```
[root@localhost]# tail /var/log/httpd/error_log
[Mon Sep 11 19:29:31 2006] [error] /akkada/var/log/exc_xml.log at
/akkada/lib/MyException.pm line 105.\nPermission denied at
/akkada/lib/MyException.pm line 104.\n
[Mon Sep 11 19:29:31 2006] [error] Can't load Perl module Runtime for server
127.0.0.1:0, exiting...
```

Check permissions to **/akkada/var/log/*** files – the Apache server user group should have RW rights to both log files. Also check **dmesg** - if there are messages such as:

```
Sep 11 19:21:37 localhost kernel: audit(1157995290.842:9527): avc: denied { append }
for pid=13842 comm="httpd" name="exc_xml.log" dev=hda3 ino=777076
scontext=root:system_r:httpd_t:s0 tcontext=root:object_r:etc_runtime_t:s0
tclass=file
Sep 11 19:21:37 localhost da kernel: audit(1157995290.846:9528): avc: denied {
append } for pid=13842 comm="httpd" name="exc_xml.log" dev=hda3 ino=777076
scontext=root:system_r:httpd_t:s0 tcontext=root:object_r:etc_runtime_t:s0
tclass=file
Sep 11 19:21:37 localhost kernel: audit(1157995290.846:9529): avc: denied { append }
for pid=13842 comm="httpd" name="exc_xml.log" dev=hda3 ino=777076
scontext=root:system_r:httpd_t:s0 tcontext=root:object_r:etc_runtime_t:s0
tclass=file
```

you need to set proper permissions in SELinux or disable SELinux.

Cannot start Apache server

```
[root@localhost~]# service httpd start
[root@localhost~]# Starting httpd: [FAILED]
[root@localhost~]# tail /var/log/ht tpd/error_log
[Fri Feb 24 23:50:04 2006] [error] Can't locate Runtime.pm in @INC (@INC contains:
/lib /usr/lib/perl5/site_perl/5.8.6/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.5/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.4/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.3/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.6 /usr/lib/perl5/site_perl/5.8.5
/usr/lib/perl5/site_perl/5.8.4 /usr/lib/perl5/site_perl/5.8.3
/usr/lib/perl5/site_perl /usr/lib/perl5/vendor_perl/5.8.6/i386 -linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.5/i386 -linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.4/i386 -linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.3/i386 -linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.6 /usr/lib/perl5/vendor_perl/5.8.5
/usr/lib/perl5/vendor_perl/5.8.4 /usr/lib/perl5/vendor_perl/5.8.3
/usr/lib/perl5/vendor_perl /usr/lib/perl5/5.8.6/i386 -linux-thread-multi
/usr/lib/perl5/5.8.6 . /etc/httpd) at (eval 5) line 3.\n
[Fri Feb 24 23:50:04 2006] [error] Can't load Perl module Runtim e for server
```

This means the Apache server does not see AKKADA or KEEENV environment variable. Check if both variables are available for the Apache server user:

```
[root@localhost]# sudo -u apache echo $AKKADA
/akkada
[root@localhost]# sudo -u apache echo $KEEPENV
PATH AKKADA
```

Check the mod_perl configuration (see the "Web GUI configuration" subchapter) if the following lines appear:

```
PerlPassEnv AKKADA
PerlPassEnv REMOTE_ADDR
```

You can also try to start the Apache server with the **/etc/init.d/httpd** script or the **apachectl** binary directly instead of using the **service httpd start** command.

Probe modules restart all the time

Try to start any probe manually:

```
[root@localhost]# cd /akkada/bin
[root@localhost]# sudo –u akkada ./np.-run.pl cpu <akkada.pl process' PID>
```

If you see error messages such as:

```
Use of uninitialized value in concatenation (.) or string at ./np -run.pl line 11.
Can't locate Entity.pm in @INC (@INC contains: /lib
/usr/lib/perl5/site_perl/5.8.8/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.7/i386-linux-thread-multi
/usr/lib/perl5/site_perl/5.8.6/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.5/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.4/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.3/i386 -linux-thread-multi
/usr/lib/perl5/site_perl/5.8.8 /usr/lib/perl5/site_perl/5.8.7
/usr/lib/perl5/site_perl/5.8.6 /usr/lib/perl5/site_perl/5.8.5
/usr/lib/perl5/site_perl/5.8.4 /usr/lib/perl5/site_perl/5.8.3
```

```
/usr/lib/perl5/site_perl /usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.7/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.6/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.5/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.4/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.3/i386-linux-thread-multi
/usr/lib/perl5/vendor_perl/5.8.8 /usr/lib/perl5/vendor_perl/5.8.7
/usr/lib/perl5/vendor_perl/5.8.6 /usr/lib/perl5/vendor_perl/5.8.5
/usr/lib/perl5/vendor_perl/5.8.4 /usr/lib/perl5/vendor_perl/5.8.3
/usr/lib/perl5/vendor_perl /usr/lib/perl5/5.8.8/i386-linux-thread-multi
/usr/lib/perl5/5.8.8 .) at ./np-run.pl line 13.
BEGIN failed--compilation aborted at ./np-run.pl line 13.
```

it means the AKKADA environment variable is not available to the AKK@DA user **akkada** or points to the wrong directory.

Chapter 3

3. Running AKK@DA

3.1. Verifying system configuration

3.1.1. Verifying AKK@DA configuration

To verify AKK@DA configuration the cfgcheck.pl tool should be used. Please be informed that this tool checks nothing more than the configuration syntax. This tool does not detect logical configuration errors.

Example 1:

```
[root@ localhost]# cd /akkada/bin
[root@localhost]# ./cfgcheck.pl

AKK@DA: configuration syntax OK
```

Output shows the configuration syntax is correct.

Example 2:

```
[root@ localhost]# cd /akkada/bin
[root@ localhost]# ./cfgcheck.pl
String found where operator expected at /akkada/etc/conf.d/Web.conf line 15, near
"'AlarmsSortOrderDefault'"
        (Missing semicolon on previous line?)
PROBLEM: syntax error in file /akkada/etc/conf.d/Web.conf (linked via
/akkada/etc/akkada.conf)
AKK@DA: configuration syntax PROBLEM (count: 1)
```

Output shows there is a problem in the /akkada/etc/conf.d/Web.conf file on line 15.

Validation
To date no validation mechanisms have been developed in AKK@DA yet. This means any values configured in the configuration files are taken without checking whether they are correct or not. This is a problem and will be fixed in the future. For now it is important to make all changes carefully and not to change anything if you don't really know what you're doing. Some options are described in this users' guide and some of them are described on the AKK@DA web site http://akkada.eu in the web based documentation.

3.1.2. Verifying the Web GUI configuration

Verifying the web GUI configuration means verifying the Apache server configuration. It can be done with the **apachectl** command.

Example 1:

```
[root@localhost]# apachectl -t
Syntax OK
```

The output shows the configuration syntax is correct.

Example 2:

```
[root@ localhost]# apachectl -t
Syntax error on line 58 of /etc/httpd/conf.d/perl.conf:
/etc/httpd/conf.d/perl.conf:58: <Location> was not closed.
```

The output shows there is a problem in the /etc/httpd/conf.d/perl.conf file on line 58.

3.2. Log files

Where are AKK@DA's log files?

AKK@DA keeps its log files in the **/akkada/var/log** directory. There are two log files: one in text format where most of the common messages are logged and one in XML format where abnormal behavior (including stack trace) messages are lo gged. By default they are named:

/akkada/var/log/exc_txt.log
/akkada/var/log/exc_xml.log.

The location and names of these files can be changed in the **/akkada/etc/conf.d/MyException.conf** configuration file:

```
{
    'TextOneLineMode' => 1,
    'StackTrace' => 1,
    'LogFileXML' => "$ENV{AKKADA}/var/log/exc_xml.log",
    'LogFileText' => "$ENV{AKKADA}/var/log/exc_text.log",
    'TimeFormat' => '',
},
```

Other options in this file shouldn't be changed.

There is a script **/akkada/bin/log.sh** which allows an easy tail of both files.

Logs rotation

There is no tool to rotate AKK@DA log files. Standard operating system tools can be used for this purpose but remember to run the **/akkada/bin/clear_log.sh** tool after log rotation to set proper rights to both log files. Without this both AKK@DA and the web GUI will stop working. The **clear_log.sh** tool can also be used without prior log rotation, but in this case remember that all messages in current logs will be lost - **clear_log.sh** deletes current log files, creates new log files and sets proper rights to these files.

Web GUI logs

In the case of the web GUI logging is a little bit more complicated. The web GUI logs to **exc_txt.log** and **exc_xml.log** files mentioned above as well as to the standard Apache error log file defined in the Apache server configuration. In order to trace errors in the web GUI all three log files should be reviewed.

Changing trace level

Trace level is configured globally in the **/akkada/etc/akkada.conf** file, option **TraceLevel**:

```
'TraceLevel' => 1,
```

Allowed values are 0 (error), 1 (warning), 2 (info), 3 (debug), 4 (internal), 5 (dbinternal). All messages with severity lower or equal to the configured value are logged. Normally for the performance reasons **TraceLevel** shouldn't be configured higher than 1.

To disable logging completely set the **LogEnabled** option in the**/akkada/etc/akkada.conf** file to 0:

```
'LogEnabled' => 1,
```

Any changes of logging options require AKK@DA to be restarted.

3.3. Starting and stopping

3.3.1. Starting and stopping AKK@DA

Start

AKK@DA has to be started from the root account. To start and stop AKK@DA use the **/akkada/etc/init.d/akkada** script.

```
[root@localhost]# /akkada/etc/init.d/akkada {start|stop|restart}
```

During the starting process AKK@DA verifies configuration with the cfgcheck.pl tool described above.

Stop

The stopping process can take up to a few minutes. This is the result of the fact that during the stopping process all probes write cached information to RRD databases and this may take a while. It is strongly recommended to verify with the **ps** command that all AKK@DA processes have been stopped before the future starting AKK@DA:

```
[root@localhost]# ps –ef | grep akkada
```

For errors and other messages during the starting and stopping processes, see AKK@DA log files.

Killing

Like any other system processes, AKK@DA processes can be killed with the **kill** command. Modules (**nm-*.pl** processes) can be killed with or without option **-9**. In the case of probes (**np-*.pl** processes) option **-9** shouldn't be used. Probes collect statistic data in their caches and when killed with option **-9**, they are unable to save cached information to RRD databases.

In the case of killing the **akkada.pl** process, all other processes will die as well. This is a raw method of stopping AKK@DA.

In the case of killing any other AKK@DA process, it will be automatically started again by the **akkada.pl** process in a short while. Killing specific AKK@DA process (except for **akkada.pl**) is the quickest way to restart the specific process, whether it's a module or a probe.

3.3.2. Starting and stopping the Web GUI

Starting and stopping the web GUI means starting and stopping the Apache server. This can be done e.g. with the **apachectl** tool:

```
[root@localhost]# apachectl {start|stop}
```

For more information,refer to the Apache documentation available at http://www.apache.org.

For errors and other messages during the starting and stopping processes, see Apache log files configured in Apache server configuration (e.g. **httpd.conf**) files.

3.3.3. Starting and stopping AKK@DA with the Web GUI

This functionality is available only when AKK@DA has already been started from command line as described in the **Starting and stopping AKK@DA** subchapter because it uses the main AKK@DA process **akkada.pl** as a request executor.

After logging on to the Web GUI click **dashboard** in top menu and go to the **system manage** tab. To start or stop all processes (except for the **akkada.pl**) use the **global action** form. To start or stop a particular process use the applicable form **modules status** or **probes status**.

Remember this is an action scheduler and scheduled actions will be completed as soon as possible but not immediately. When action is scheduled but not executed, message **in progress** is presented instead of the pop-up menu.

modules status:

	name	process count	expected process count		action
	actions_broker	1	1	OK	--- select ---
	actions_executor	1	1	OK	--- select ---
	available	1	1	OK	--- select ---
	db_watch	1	1	OK	--- select ---
	discover	1	1	OK	--- select ---
	icmp_monitor	1	1	OK	--- select ---
	job_planner	1	1	OK	--- select ---
	status_calc	1	1	OK	--- select ---
	sysstat	1	1	OK	stop in progress
	tree_cache	1	1	OK	--- select ---

OK

Table 3. Starting and stopping AKK@DA with the Web GUI

3.4. Accessing the Web GUI

To access the AKK@DA Web GUI use any supported web browser (see chapter 1 for the list of supported web browsers). The Web GUI is available under the IP address of the AKK@DA server. Initial log on screen:

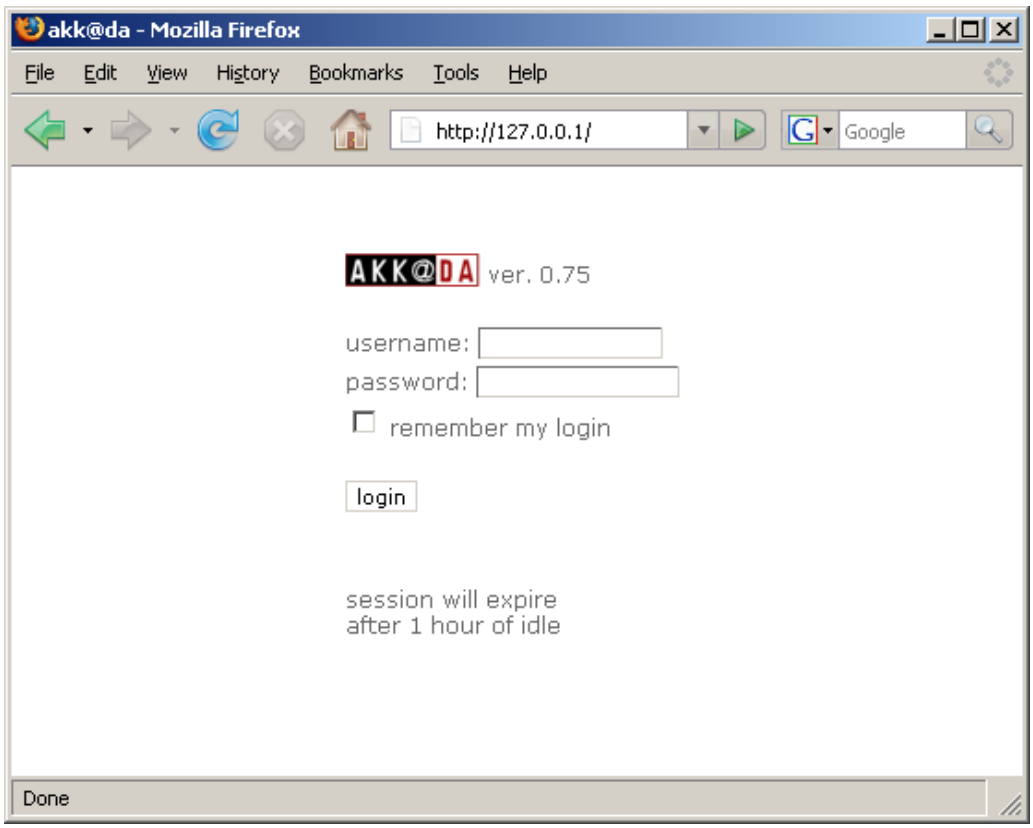


Figure 1. Log on screen

Initially two users exist:

user name	password	Role
operator	operator	limited read-only access
yoda	upfs	full read-write access to all options

Table 4. Default user names and passwords

For more information about managing users and their rights in AKK@DA , see chapter **Managing security on AKK@DA**.

Chapter 4

4. The core of AKK@DA

4.1. Overview

The core of AKK@DA is a group of scripts. Their function is to manage tests, calculate statuses, cache objects between probes and the web GUI, deal with notifications, etc. Generally all modules which are not probes are described in this chapter (for probes, see chapter **Services**, subchapter **Probes**).

All modules described in this chapter have their own configuration files. After making any change of a specific configuration file, the specific module has to be restarted to make this change work.

4.2. Modules

4.2.1. nm-actions_broker.pl

Overview

The **actions broker** module is a part of the notifications subsytem of AKK@DA and is responsible for managing notifications. If notification for the specific entity is configured and this entity status has changed, the probe responsible for testing this specific entity raises a notification regardless of the defined notification conditions (for the performance reasons probes don't analyze notification conditions). Then the **action broker** module analyzes whether the notification meets defined notification conditions. If it does, the notification task is prepared for the **actions executor** module, otherwise notification is ignored.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

If the notification functionality is not used, it is recommended to keep this module stopped, otherwise it should always be kept running. When **actions broker** is stopped, notifications are not processed.

Configuration

Configuration of the **actions broker** module is kept in file **/akkada/etc/conf.d/ActionsBroker.conf**:

```
{
    'Period' => 1,
    'ActionsDir' => "$ENV{AKKADA}/var/actions",
}
```

Name	Description
Period	Sleep time between subsequent notification processing (seconds); by default 1 second; it should be a low value; longer period may cause delays in notification processing
ActionsDir	Directory where probe modules create temporary notification files which are processed by the **actions broker** module

Table 5. actions broker configuration parameters

4.2.2. nm-actions_executor.pl

Overview

The **actions executor** module is a part of the notification subsytem of AKK@DA. It is responsible for sending notifications only. It doesn't analyze any notification conditions, but only sends what has been prepared by the **actions broker** module.

Restart

This module can be safely restarted any time needed without information being lost

Stop

If the notification functionality is not used, it is recommended to keep this module stopped, otherwise it should always be kept running. When **actions executor** is stopped, notifications are not sent.

Configuration

The main configuration of the **actions executor** module is kept in file **/akkada/etc/conf.d/ActionsExecutor.conf**:

```
{
    'Period' => 1,
    'Modules' => {
        'mail' => do "$ENV{AKKADA}/etc/conf.d/ActionsExecutor/mail.conf",
        'GTalk' => do "$ENV{AKKADA}/etc/conf.d/ActionsExecutor/GTalk.conf",
    },
}
```

Name	Description

Period	Sleep time between subsequent notification sending (seconds); by default 1 second; it should be a low value; a longer period may cause delays in notification sending
Modules	Links to files with configurations of the specific notification mechanisms; for more information regarding these linked configuration files, see chapter **Managing notifications with AKK@DA**

Table 6. Actions executor configuration parameters

4.2.3. nm-available.pl

Overview

The **available** module checks the reachability of all host entities monitored by AKK@DA (and group entities if they have IP addresses configured) with ICMP protocol. The role of this module is to quickly detect if the specific entity is unreachable when it is not available on the network.

The speed of unreachability detection is essential from the perspective of AKK@DA. When the host becomes unavailable on the network all tests performed by AKK@DA against this host fail with timeouts. Timeoutes are very long if compared with test times when the host is available. These long timeouts could degradate AKK@DA performance in the case of the host unreachability. The **available** module solves that problem. When the host unreachability is detected, the **available** module sets this host status to UNREACHABLE and changes the status of all services of this host to UNKNOWN. From that time services of this host are not tested by AKK@DA until this host is available on the network again.

To assure quick detection of host unreachability, the **availabe** module checks all monitored hosts constantly. It uses **fping** software to perform these ICMP tests.

The UNREACHABLE status is detected based on the policy configured (see section **Configuration**).

☝ The **available** module must be disabled if **available2** module is enabled.

When to use?

The available module can be used as a replacement of the **available2** module. Generally **available2** is a better choise. It's more modern and includes inference engine which informs an operator what is a root cause of a detected network fault. However, **available2** consumes more CPU time and has some requirments which should be met otherwse it doesn't work properly (see subchapter **available2** in this chapter). Use module **available** when:

- AKK@DA has limited ICMP access to monitored hosts (**available2** needs ICMP access to as many as possible IP addresses of monitored network interfaces; **available** works fine when it has ICMP access only to primary host

IP address; e.g. if AKK@DA can ping only management router's interface, not all router's interfaces, use **available**),

- there are overlapping IP addresses on monitored network ,
- AKK@DA's server has highly utilized CPU and there are no resources for other CPU consumming process.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This module should always be running. If stopped, hosts unreachability is not detected, UNREACHABLE alarms are not raised and the performance of the whole AKK@DA is degraded.

Configuration

Configuration of this module is kept in file **/akkada/etc/conf.d/Available.conf**:

```
{
    'ErrMsg' => 'not reachable through ICMP',
    'fping' => [
      '/usr/local/sbin/fping',
      '-q',
      '-C'
    ],
    'PingCount' => 4,
    'Period' => 1
}
```

Name	Description
ErrMsg	Error message which is shown by the web GUI when the specific host status is UNREACHABLE
fping	Full path to the **fping** binary file and options for **fping**
PingCount	Number of ICMP echo requests sent during a test (number); UNREACHABLE status is detected if replies for all these echo requests are missing; default value is 4
Period	Sleep time between subsequent tests (seconds); by default 1 second; it should be a low value; a longer period may cause delays in detecting availability of entities on the network

Table 7. Available configuration parameters

4.2.4. nm-available2.pl

Overview

The **available2** module checks with ICMP protocol the reachability of all IP addresses related to host entities monitored by AKK@DA. The role of this module is to quickly detect if the specific entity is unreachable when it is not available on the network. In an opposite to **available** module, it works based on the monitored network structure

model. That allows it to detect not only host unreachability, but also unreachbillity of whole parts of the network. In that case the **available2** module detects also a root cause of detected network unreachbility. Alarms raised by module **available2** contain visualisation on the afected part of the monitored network.

To assure quick detection of host and network unreachability, the **availabe2** module checks all IP addresses of monitored hosts (IP addresses of all host's interfaces) all the time. It uses **fping** software to perform these ICMP tests.

The UNREACHABLE status is detected based on the policy configured (see section **Configuration**).

☝ The **available2** module must be disabled if **available** module is enabled.

When to use?

The **available2** module can be used as a replacement for the **available** module. Generally **available** is a worse choice. It's older, detects only host unreachbility and it's not suggested for using. However, **available2** consumes more CPU time and has some requirments which should be met otherwse it doesn't work properly. Use module **available2** only when:

- AKK@DA has ICMP access to all or almost all IP addresses of monitored network interfaces; otherwise **available2** will not be able to correctly detect network faults,
- there are NO overlapping IP addresses on monitored network,
- AKK@DA's server has some free CPU resources.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This module should always be running. If stopped, host unreachability is not detected, UNREACHABLE alarms are not raised and the performance of the whole AKK@DA is degraded.

Signals

This module supports the following signals:

Signal	Description
HUP	Module reloads configuration.
USR1	Increase logging level +1
USR2	Decrease logging level -1

Configuration

Configuration of this module is kept in file **/akkada/etc/conf.d/Available2.conf**:

```
{
    'ErrMsg' => 'not reachable through ICMP',
```

```
    'fping' => [
      '/usr/local/sbin/fping',
      '-q',
      '-C'
    ],
    'PingCount' => 4,
    'Period' => 1,
    'GraphDebug' => 0,
    'GraphDebugPath' => '/akkada/var/rrd_graph_tmp',
    'LowLevelDebug' => 0,
    'NetDesc' => '/akkada/etc/netdesc.conf',
    'ifconfig' => '/sbin/ifconfig',
    'ifconfig_addr' => 'inet addr:',
    'ifconfig_mask' => 'Mask:',
    'CheckingFlagsDir' => "$ENV{AKKADA}/var/av2",
    'DOTranksep' => '1.8',
    'DOT' => 'fdp',
    DisabledIPAddr => {
    },
    PreferredNetworks => {
    }
}
```

Name	Description
ErrMsg	Error message which is shown by the web GUI when the specific host status is UNREACHABLE
fping	Full path to the **fping** binary file and options for **fping**
PingCount	Number of ICMP echo requests sent during a test (number); UNREACHABLE status is detected if replies for all these echo requests are missing; default value is 4
Period	Sleep time between subsequent tests (seconds); by default 1 second; it should be a low value; longer period may cause delays in detecting availability of entities on the network
GraphDebug	Enables/disables generating image with graph

	representing network model after any change of this model (detecting any IP address state unreachable/reachable change); useful when something goes wrong; makes **available2** work much slower and needs more CPU time; 0 means disabled, 1 means enabled; generated image name is always **graph.png**
GraphDebugPath	Directory, when debug image **graph.png** is generated
LowLevelDebug	Enabling/disabling low level debugging messages; 0 means disabled, 1 means enabled;
NetDesc	Full path to the file where subnet descriptions are kept. **Available2** always saves a list of monitored subnets into this file; to describe subnets, this file should be manually edited or by using the web GUI, top menu, button **tools**, **subnet descriptions**; these descriptions are shown on images related with alarms raised by **available2**
ifconfig	Full path to the **ifconfig** tool. Available2 uses this tool to detect AKK@DA's self IP addresses
ifconfig_addr	String used for selecting IP addresses
ifconfig_mask	String used for selecting network masks
CheckingFlagsDir	Directory where checking flags are kept
DOT	Name of the **Graphviz** filter used for generating pictures of network; For more details, see Graphviz web site http://www.graphviz.org; available options: neato, twopi, dot, circo, fdp

DisabledIPAddr	Network structure model build by **available2** is unable to recognize the real ways of passing traffic; it focuses only on connections between hosts; for that reason sometimes it's needed to force **available2** to ignore some of IP addresses; e.g. syntax:
	`DisabledIPAddr => {`
	`  '10.0.5.252' => 1,`
	`  '10.0.5.253' => 1,`
	`},`
	this is especially needed when there are dedicated management networks where all network devices have their management interfaces; in that case all management IP addresses should be disabled by the **DisabledIPAddr** key

PreferredNetworks	

Table 8. Available2 configuration parameters

4.2.5. nm-db_watch.pl

Overview

The **db watch** module monitors the number of entities in the database and informs all other processes if the number of entities in the database has changed.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This module should always be kept running. If stopped AKK@DA doesn't monitor any newly discovered entities.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/DBWatch.conf** file:

```
{
    'Period' => 30,
},
```

Name	Description
Period	Sleep time between subsequent tests (seconds); by default 30 seconds

Table 9. db_watch configuration parameters

4.2.6. nm-discover.pl

The discover module is described in chapter **Services**, subchapter **Discovery process**.

4.2.7. nm-icmp_monitor.pl

Overview

The **icmp monitor** module collects ICMP statistics (packet delays, loss, jitter). This module is something between the core module and the probe module. From the logical perspective this is another probe module. But from the performance perspective it works in a completely different way than other probe modules.

The **icmp monitor** collects ICMP statistics by sending 20 ICMP echo requests to all IP addresses associated with **nic** and **icmp_monitor** entities (see chapter **Services**,

subchapter **Probes**). Results are saved in RRD databases. This module also raises alarms if a packet loss is detected or delays exceed defined thresholds.

In the case of the **nic** entity, ICMP statistics are collected by **icmp monitor** automatically if only the specific **nic** has an IP address.

Restart

This module can be restarted any time needed but information cached in the module memory may be lost.

Stop

This module should always be kept running if collecting ICMP statistics is needed.. If stopped AKK@DA doesn't collect ICMP statistics.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/ICMPMonitor.conf** file:

```
{
    'ThreadsCount' => 3,
    'Period' => 1,
    'StatusDir' => "$ENV{AKKADA}/var/icmp_status",
    'fping' => [
      '/usr/local/sbin/fping',
      '-q',
      '-C'
    ],
    DefaultLostThreshold => 10,
    DefaultDelayThreshold => 0.05,
    DisableUnreachableIPAtFirstTimeCheck => 1,
}
```

Name	Description
ThreadsCount	Number of threads of the **icmp monitor** process (number); by default 3; one thread should be able to monitor effectively approximately 100 hosts, but it also depends on the response times of these hosts
Period	Sleep time between subsequent tests (seconds); by default 1 second
StatusDir	Full path to the directory where temporary flag files are kept; this must be a separated directory for this module only
fping	Full path to the **fping** binary file and options for **fping**
DefaultLostThreshold	See chapter **Services**, subchapter **Probes**, **nic**
DefaultDelayThreshold	See chapter **Services**, subchapter **Probes**, **nic**
DisableUnreachableIPAtFirstTimeCheck	1 = disable collecting ICMP statistics if the IP address doesn't respond during the first test; 0 disables this mechanism

Table 10. icmp_monitor configuration parameters

4.2.8. nm-job_planner.pl

Overview

The **job planner** module manages tests performed by probes. For each type of entity it compares their count with the count of probe processes of this type and divides entities of the specific type between all probe processes of this specific type equally.

Restart

This module can be safely restarted any time needed without information being lost. Each restart of this module starts the initialization process of all probe processes which means AKK@DA performance is temporarily degraded.

Stop

This module must always be kept running and if it's stopped AKK@DA doesn't test anything.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/JobPlanner.conf**: file:

```
{
    'Period' => 60,
},
```

Name	Description
Period	Sleep time between subsequent checks if the entities' count has changed (second); by default 60 second

Table 11. job_planner configuration parameters

4.2.9. nm-status_calc.pl

Overview

The **status calc** module calculates statuses of host and group entities based on statuses of all their children entities. The **location tree** structure (see chapter **Configuring services with AKK@DA**, subchapter **Entities basics**) is used for calculating statuses.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This module must always be kept running If stopped AKK@DA doesn't calculate statuses of group and host entities and information presented by the web GUI may be wrong.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/StatusCalc.conf**: file:

```
{
    'ThresholdHigh' => 40,
    'StatusCalcDir' => "$ENV{AKKADA}/var/status_calc",
    'ThresholdMed' => 17,
    'Period' => 5
}
```

Name	Description
ThresholdHigh	Threshold high used during status calculation; modifying it changes the policy of the status calculation (%); shouldn't be modified; by default 40 %
StatusCalcDir	Full path to the directory where temporary flag files are kept; this must be a separated directory for this module only
ThresholdMed	Threshold medium used during status calculation; modifying it changes the policy of the status calculation (%); shouldn't be modified; by default 17 %
Period	Sleep time between subsequent status calculations (seconds); by default 5 seconds; shouldn't be too long, otherwise information presented by the web GUI may be not up-to-date

Table 12. status_calc configuration parameters

4.2.10. nm-sysstat.pl

Overview

The **sysstat** module collects AKK@DA performance data regarding the performance of testing entities. It collects global and detailed histograms of test periodicity, number of log entries, global number of entities in a specific state, etc. All these statistcis are available on the web GUI at in the **dashboard** section. Additionaly a part of the **sysstat** module is **freshness guard**. The freshness guard function is to raise an alarm if a specific entity is not tested frequently enough (e.g. if an entity check period is set to 60 seconds, but it is tested every 12 minutes, it will raise an alarm because this is unwanted situation and probably a symptom of degradated AKK@DA system performace).

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This is an optional module. If stopped, AKK@DA performance statistics are not collected and most notably **histograms** on the web GUI (part of the **dashboard** section) are not available. **Freshness guard** also doesn't work so the automatic ensuring that AKK@DA meets the checking frequency policy is disabled.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/SysStat.conf**: file:

```
{
    'Period' => 60,
    'LastCheckHistogramFilePrefix' => "last_check_histogram",
    'LastCheckHistogramDir' => "$ENV{AKKADA}/var",
    'FreshnessGuardEnabled' => 1,
    'FreshnessStartCalcAfter' => 600,
    'FreshnessStaleAlarmLevel' => _ST_DOWN,
    'FreshnessThreshold' => 5,
}
```

Name	Description
Period	Sleep time between subsequent AKK@DA performance statistic calculations (seconds); by default 60 seconds
LastCheckHistogramFilePrefix	Prefix used for histogram files
LastCheckHistogramDir	Full path to the directory where histograms are stored
FreshnessGuardEnabled	0/1 = disable/enable the freshness guard; by default 1
FreshnessStartCalcAfter	Number of seconds after which the **freshness guard** functionality will be activated; by default 600; it shouldn't be less than 300 seconds, 600 seconds usually is a safe value; this mechanism exists to give the probe processes time for testing entities and to avoid false alarms
FreshnessStaleAlarmLevel	Alarm level which will be raised when a stale entity is detected; by default _ST_DOWN
FreshnessThreshold	Based on this value **freshness guard** decides if an entity is fresh or stale; if the entity last check time is older than the entity checking period multiplied by **FreshnessThreshold**, the stale state is detected

Table 13. sysstat configuration parameters

4.2.11. nm-top.pl

Overview

The **top** module generates top utilization reports for CPU, RAM, HDD and NIC entities. These reportas are avilable on the web GUI, **dashboard**, the **top** tab. These reports show the list of the most utilized services on a monitored network.

Restart

This module can be safely restarted any time needed without information being lost.

Stop

This is an optional module. If stopped, the **top** tab on the web GUI (part of the **dashboard** section) is not available.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/Top.conf**: file:

```
{
    'TopDir' => "$ENV{AKKADA}/var/top",
    'Period' => 60,
    'ListSize' => 10,
    'Expire' => 300,
    'DisplayColumns' => 2,
},
```

Name	Description
TopDir	Directory where the utilization data for the **top** module is stored by probe processes
Period	Sleep time between subsequent checks (seconds); by default 1 second
ListSize	Size of the list of the most utilized services which is presented on the **top** tab; by default 10;
Expire	Maximum age of the utilization data (seconds); by default 300; it shouldn't be less than 300
DisplayColumns	Number of columns used on the web GUI by the **top** tab for arranging tables in the report

Table 14. top configuration parameters

4.2.12. nm-tree_cache.pl

Overview

The **tree cache** module stores whole **location tree** structure of monitored environment as a cached data structure. This cache is essential for the web GUI - without it the web GUI doesn't work

Restart

Restarting of this module affects the web GUI only. The web GUI doesn't work until the **tree cache** module ends its initialization process which may take a few minutes (depending on the number of monitored entities).

Stop

This module must always be kept running. If stopped the AKK@DA's web GUI doesn't work, although the rest of the AKK@DA system works normally. In other words, AKK@DA keeps monitoring without that process, but it is not possible to see the result of monitoring on the web GUI without this process.

Configuration

Configuration of this module is kept in the **/akkada/etc/conf.d/Web.conf** file:

```
{
[...]
    TreeCacheDir => "$ENV{AKKADA}/var/tree_cache",
    TreeCachePeriod => 1,
[...]
}
```

Name	Description
TreeCacheDir	Full path to the directory where the cache is kept; usually the cache needs approximately a few MB of disk space; it is recommended to keep this cache in the RAM disk for better performance
TreeCachePeriod	Sleep time between the subsequent tests for changes of

	the data structure (seconds); by default 1 second; should be short, otherwise information presented by the web GUI may be not up-to-date

Table 15. tree_cache configuration parameters

Chapter 5

5. Managing AKK@DA

5.1. AKK@DA health

There are two places where the AKK@DA system health status is presented. They are available at **dashboard** (to access dashboard click the **dashboard** button in top menu), tabs **general** and **system status**.

To view correct information at **dashboard** always restart the Apache server with the web GUI after changes have been made in the **/akkada/etc/conf.d/System.conf** file.

General: system

Tab **general** presents a few tables with information about the state of AKK@DA.

Table **system** shows the number of enabled and disabled probes and modules, the total number of their running instances, the total number of down processes and other processes with problems as well. Column **down** presents the total number of down processes which means some of the expected module or probe processes are not started. A situation like that occurs when some of the processes were manually downed for any reason. If they weren't this probably means there is a fatal software error and AKK@DA does not work well in full or in part (depending on what is downed). More details about these kinds of problems can be found at the **system status** tab. Generally AKK@DA log files should be reviewed to discover the reason of a problem.

system:

	enabled	disabled	up	down	problems
modules	10	0	10	0	0
probes	30	7	37	0	0

Table 16. Dashboard, tab general, table system

The **problem** column presents the total number of problems within processes. These problems mean a higher number of processes than expected. Also in the case of

probes these problems can mean the number of entities processed by probes does not equal the total number of entities defined in a configuration. These kinds of problems are reported often and this is normal (e.g. when new services were discovered and probes didn't have enough time to start processing them). If a situation like that keeps occurring then probably something is wrong and AKK@DA should be restarted.

General: entities count

The entities count table presents important information about the internal cache state. The internal cache is a place of sharing information between probes and the web GUI. Normally the total number of entities should equal the total number of cached entities. They can be different only during the discovery process in a situation when new entities have been discovered and added to the configuration but the cache hasn't been updated yet. If the difference between these two numbers persists longer than a few minutes then probably there is an error in the cache data structure and it's necessary to fix the cache manually. The procedure of fixing the internal cache is described below.

entities count:		
total	4733	
cached	4733	
monitored	4687	99.03%
not monitored	46	0.97%

Table 17. Dashboard, tab general, table entities count

Fixing the internal cache

Known symptoms of errors in the internal cache: the number of entities in the cache is different than the number of entities in the configuration, the web GUI failed to display anything or failed to display information about specific entities, information presented by the web GUI is not up to date.

In all cases the only solution is to remove the internal cache. After the internal cache has been removed AKK@DA recreates it. To do this first the **nm-tree_cache.pl** process must be stopped and then all files and directories must be removed from the directory where the cache is located (by default the internal cache is kept in the **/akkada/var/tree_cache** directory; this directory is configured in the **/akkada/etc/conf.d/Web.conf** file, the **TreeCacheDir** option). You can do this e.g. with the following command (works on RedHat and Fedora):

```
[root@ localhost]# killall nm-tree_cache.pl && rm –rf \
/akkada/var/tree_cache/*
```

After doing this you have to wait till the **akkada.pl** process starts the **nm-tree_cache.pl** process (see AKK@DA log files or system processes with the **ps** command). When the **nm-tree_cache.pl** process has been restarted you have to wait for a while till the internal cache is rebuilt by the **nm-tree_cache.pl** process (this can take a few seconds or a few minutes – depending on the number of entities monitored by AKK@DA; during rebuilding the internal cache the **nm-tree_cache.pl** process consumes all the available CPU time, so it is easy to see when it's done with the **top** command).

General: mysql status

The **mysql status** table presents the MySQL server statistics. For more information about these statistics, refer to the MySQL documentation (http://www.mysql.org/) .

5.2. AKK@DA efficiency

General AKK@DA efficiency information is available at **dashboard**, the **histograms** tab (to access the dashboard click the **dashboard** button in top menu). Detailed information about AKK@DA efficiency is available for every service.

Histograms

AKK@DA checks all services periodically. By default services are checked every minute, but this is not a strict period inbetween test. This is the period between scheduling the subsequent tests. If AKK@DA works efficiently the real period between subsequent tests is between 60 and 75 seconds. If AKK@DA performance problems occur or when monitored nodes are overloaded and take a long time to answer the real period between subsequent tests may increase even to a few minutes, what means that the potential alarms may be raised after a few minutes instead of 75 seconds as expected.

Histograms are a simple way of obtaining information about AKK@DA meeting (or not) configured periods between subsequent tests. Below there is an example histogram for the snmp_generic probe:

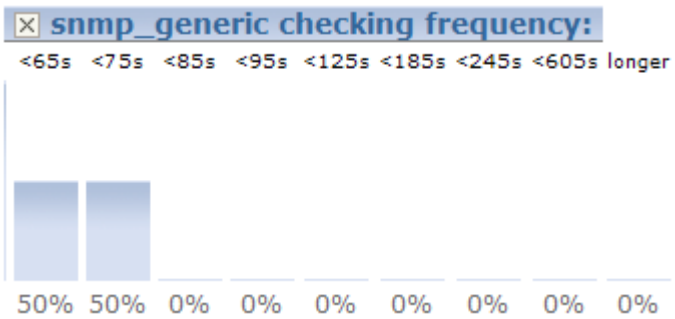


Table 18. snmp_generic histogram example 1

This histogram shows that 50% of snmp_generic type services are checked every 6 5 seconds and 50% of them are checked every 75 seconds. It means that AKK@DA's performance is fine. In such case no action is required. Look at the next example:

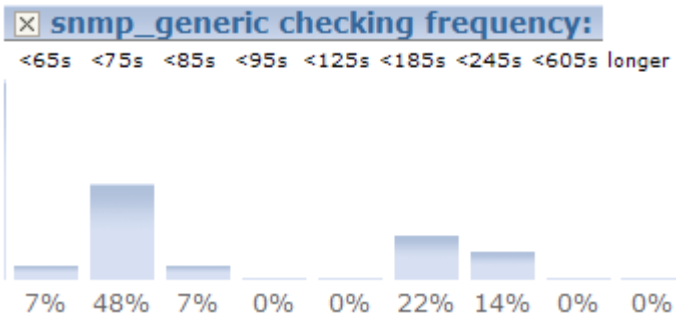


Table 19. snmp_generic histogram example 2

This histogram shows there are 36% of snmp_generic type services whose period between subsequent tests is longer than 3 minutes what means that AKK@DA's performance is degraded (unless you configured the checking period to be 3 minutes!). This situation suggests there are performance problems.

Efficiency information per service

For any monitored service, at the **stat** tab there is available a graph which shows the period between subsequent tests in the case of that particular service. This graph is at the end of the list of graphs and is called **delta**. E.g..:

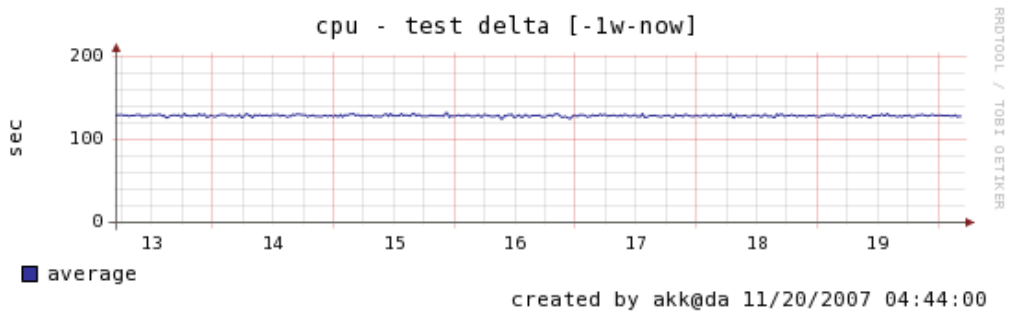


Table 20. Delta graph example

This graph shows that a specific **cpu** service of one of nodes is tested approximately every 130 seconds. If the checking period is configured to 120 seconds this is fine. If the checking period is configured to 1 minute this means we have a problem with testing efficiency.

Number of probe instances

One instance of probe can effectively test only a limited number of services. Probes are regular processes and they don't use threads. For that reason one probe can efficiently test around 200-300 services (if the checking period is configured to 1 minute). When a probe is overloaded symptoms described in sections **Histograms** and **Efficiency information per service** occur. In such situation the number of specific type probe instances should be increased. This can be configured in the **/akkada/etc/conf.d/System.conf** file. After changes in this file have been made the whole AKK@DA system must be restarted and as well as the web GUI.

To check how many services are being tested by a single specific probe process, go to **dashboard** in top menu, the **system status** tab, the **probes status** table. The **entities count in probe** column shows that information.

DoS

Increase the number of probe instances only when really needed. When too many probe instances operate at the same time the efficiency of the system can also deteriorate. This happens because too many probes test different services of the same node at the same time and as a result the tested node may start to respond slower. In an extreme situation a node may be killed in the same way as when using DoS attack.

High load average

When the AKK@DA's server has high load average utilization there usually is a problem with the disk efficiency because of saving statistic information to RRD database files. AKK@DA collects a lot of statistic and performance data and all this data is kept in the RRD databases. The RRD database is a specific type of database which must be supplied with data every defined amount of time. This means that if you monitor e.g. 3000 services every minute, every second information about 50 (3000/60 = 50) services is updated, which means every second 50 RRD database files are updated. This is a lot to do for hard drives and this may cause high load average. To relieve disks AKK@DA probes cache statistic and performance information including time stamps in memory. This is done in order to save a lot of data to RRD database files using one update what allows more seldom updating of RRD database files.

The way how the above described caching mechanism works is configured in the **/akkada/etc/akkada.conf** file. When the AKK@DA's server has high load average it may be helpful to increase **RRDCacheMaxEntries** and **RRDCacheMaxEntriesNotOK** parameter values. Any changes to these parameters require AKK@DA to be restarted. Default settings:

```
'RRDCacheMaxEntries' => 16
'RRDCacheMaxEntriesNotOK' => 8,
'RRDCacheMaxEntriesRandomFactor' => 4,
```

RRDCacheMaxEntries defines the amount of subsequent collected information kept in the cache when the service status is **OK**. This means when the status of the specific service is OK, information collected during 15 subsequent tests is kept in the cache and when 16th test is done all information collected during last 16 tests is saved to the RRD database file. When **RRDCacheMaxEntries** is set to **1** information is not cached. Value 0 is not allowed. There is no upper limit for this value. When information is in the cache and is not yet saved to the RRD database file, at the web GUI you will see lack of information regarding currently cached information. E.g.:

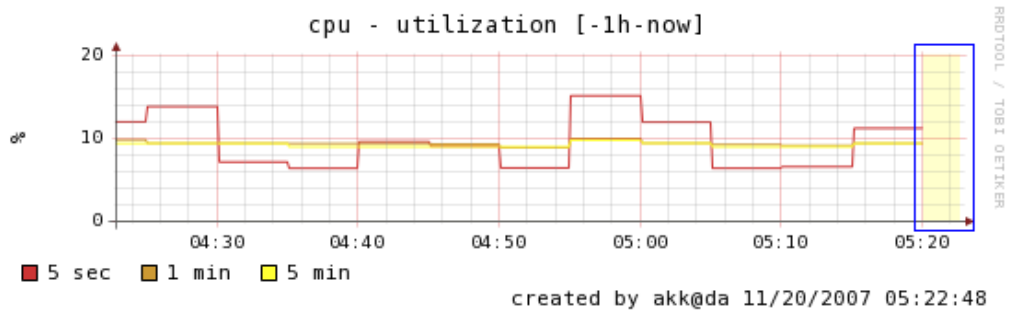


Figure 2 RRD cache example

In the graph above, the yellow inside of the blue rectangle shows lack of information. The graph was created at 05:22:48, but shows there is missing information between 05:20:00-05:22:48. This missing information is actually kept in the cache by the probe process which tests this service.

RRDCacheMaxEntriesNotOK means the same as **RRDCacheMaxEntries** except for the fact that its value is used when the specific service status is **other than OK**. This is because when there is an alarm regarding a specific service reported by AKK@DA, an administrator or user may want to see as up-to-date as possible statistic/performance information, so in this situation it is reasonable to save statistic/performance information more often.

RRDCacheMaxEntriesRandomFactor is the upper limit of the range between 0 and this value. Information is saved from the cache to RRD database files when RRDCacheMaxEntries/RRDCacheMaxEntriesNotOK amount of data is collected plus a random number from the range **0-RRDCacheMaxEntriesRandomFactor**. This protects hard drives from simultaneous mass data savings.

CPU

The **vmstat** tool is the easiest way to check the CPU usage and see if the AKK@DA server is overloaded or not:

```
[root@localhost]$ vmstat 1 100
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 2  1    244  22072  38536 3122000    0    0     0    16    1     1 63  2 30  5
 3  0    244  22152  38536 3122000    0    0     4   788 1934  1312 64  3 25  9
 5  0    244  22024  38536 3122000    0    0     4     0 1650  1424 88  3  9  0
 5  1    244  21896  38536 3122000    0    0     0  5848 1730  1197 89  3  6  1
 6  2    244  22024  38536 3122000    0    0     0   680 2046  1047 95  4  0  1
 4  0    244  22152  38544 3121992    0    0     0  1372 1788  1261 90  2  6  2
 2  0    244  22280  38544 3121992    0    0     0     0 1542  1483 49  2 49  0
 2  0    244  21832  38544 3121992    0    0     0     0 1441  1361 41  3 56  0
 1  2    244  21832  38544 3121992    0    0     0  3812 1586  1762 31  1 51 16
 3  2    244  22008  38544 3121992    0    0     0  2212 1950  1424 81  3  6 10
 9  2    244  21688  38544 3121992    0    0     0  1648 1731  1173 27  2 50 20
 8  1    244  21944  38544 3121992    0    0     0  1764 1907   842 96  3  0  1
10  1    244  21944  38544 3121992    0    0     0  1832 1992  1133 94  3  2  1
 3  2    244  22080  38544 3121992    0    0     0  5968 1917  1439 78  2  9 11
 2  2    244  22144  38544 3121992    0    0     0   440 1775  1815 35  2 34 29
 3  0    244  22208  38544 3121992    0    0     0  1536 1734  1671 54  2 31 13
 4  1    244  22048  38544 3121992    0    0     0  1604 1771  1551 83  3 11  3
 9  1    244  22048  38544 3121992    0    0     0  1856 1838  1024 94  4  1  1
```

```
 8  2    244  21856  38544 3121992    0    0     0  6576 2036   850 90  5  4  1
 8  2    244  22048  38544 3121992    0    0     0   524 2039   908 94  3  1  2
10  2    244  22056  38548 3121988    0    0     0  2684 1962  1210 72  2 19  7
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 3  0    244  22120  38548 3121988    0    0     0  1336 2113  1489 85  2 10  2
 2  1    244  22056  38548 3121988    0    0     0  1320 1806  1660 77  3 19  1
 2  2    244  22056  38548 3121988    0    0     0  4896 1959  1689 71  3 21  6
 5  2    244  22248  38548 3121988    0    0     0  1192 2064  1505 86  3  8  3
 1  2    244  22184  38556 3121980    0    0     0  2636 1864  1700 32  2 49 18
 7  0    244  22184  38556 3121980    0    0     0  1260 1959  1313 89  2  5  3
 9  1    244  22184  38556 3121980    0    0     0   980 1646  1231 92  3  5  0
```

When CPU high usage occurs there are usually 4 options to fix this issue. Generally, if there are any other applications running on the AKK@DA's server, they should be moved to another machine. AKK@DA should have a dedicated server. If there are no other applications running, only AKK@DA, it's good to review the number of probes running – maybe there are unneeded processes running (see the **Number of probe instances** subchapter in this chapter). Next step is to review monitored services and disable unneeded but monitored services which should reduce CPU usage. The last option to fix high CPU usage is increasing service checking periods. This makes services be tested more seldom and AKK@DA needs less CPU resources. If all of the above mentioned methods do not work, the only way to handle this problem is hardware upgrade or exchange.

High CPU usage caused by too few CPU resources can be detected with **vmstat**. The first column (**Procs**, **r**) shows the number of processes waiting for run time. If this value is continuously much higher than the number of CPU cores available on the specific hardware platform you have overloaded CPU. E.g. at the **vmstat** listing above, if this is on the server with 4 double core CPU, everything is fine. But if this occurs on the server with 1 double core CPU, this server is completely overloaded.

RAM and the virtual memory (swap)

Overloaded RAM problems usually make high usage of the virtual memory (swap) and cause high load average. This can be easily detected with **vmstat,** both columns **swap** (see subchapter **CPU** in this chapter). If everything regarding RAM is fine both columns **si** and **so** should have zeros. Otherwise there is too little RAM available for processes. Each AKK@DA process, including each instance of probe needs approximately 48-64 MB of RAM. If RAM is overloaded it's worth reviewing the number of running probes – maybe there are unneeded processes (see the **Number of probe instances** subchapter in this chapter). The **top** Unix tool can be used to detect which processes are the top consumers of RAM. If there is no option to reduce the number of running processes, the only way to handle this problem is RAM upgrade.

It may be worth checking if there are memory leaks in AKK@DA processes. This can be done by restarting AKK@DA and checking RAM usage by AKK@DA processes with the **top** tool. Repeat checking RAM usage a few times within a few hours to detect which processes use more and more RAM. If a memory leak is detected, there is probable a software error and it should be reported to the support of AKK@DA.

Chapter 6

6. Configuring devices with AKK@DA

6.1. Preparing the network and hosts for configuration

AKK@DA uses a couple of protocols for monitoring specific services such as SNMP, ICMP, SSH, pure TCP, DNS, HHTP, SSL, NTP, etc. The two most important are SNMP and ICMP. This chapter describes suggestions relating to them.

6.1.1. ICMP

It is very important for performance reasons to allow AKK@DA ICMP echo access to monitored hosts for quick detection of their availability. This protocol is also used to collect latency, jitter and lose statistics.

Disabling ICMP availability checks
Sometimes it's not possible to allow AKK@DA ICMP access to the monitored host (e.g. due to a company's security policy). In this case option "**disable ICMP availability check**" at the "**add node**" form (the "**add node**" form is described later in this chapter) has to be set to 1, but be informed the whole AKK@DA performance can be highly affected when the host with option "disable ICMP availability check" set to 1 is not available on the network. **It is recommended to avoid using this option without an important reason.**

At present the availability checking mechanism is able to use the ICMP protocol only.

6.1.2. Firewall access lists

The easiest way to give AKK@DA access to the monitored network is to allow it full access to the whole monitored network, but this is usually against the security policy of most companies.

If full access is not allowed, AKK@DA's server should be allowed to access monitored hosts by ICMP and UDP/161 (SNMP).

6.1.3. SNMP

The SNMP protocol is the most important protocol used to test the monitored network and to collect statistics. It is strongly recommended to allow the AKK@DA's server an SNMP access to monitored hosts otherwise AKK@DA has only a limited ability to test them.

There are three versions of the SNMP protocol used at present: 1, 2c and 3. All of them are supported by AKK@DA. To configure a device in AKK@DA it is needed to know some very basic concepts regarding SNMP access control. For security reasons it is strongly recommended to use SNMP version 3 instead of 1 or 2c if possible. The SNMP version 1 and 2c are believed to be insecure protocols.

The **SNMP agent** is software located on a monitored device in order to present information about this device for a monitoring system.

SNMP versions 1 and 2c basics

The access control mechanism of SNMP version 1 and version 2c is based on the password called "**community string**". Community string is sent through the network without any encryption. The same community string has to be configured on the monitored device and in the monitoring system configuration to make SNMP work. There are two types of community strings: **read-only** and **read-write**. It depends on which one is used by the monitoring system and whether the monitoring system has access to write information to the monitored host or not. **AKK@DA needs only read-only access to the monitored host**. When you use SNMP version 1 or 2c it is recommended for the security reasons not to configure read-write community string on devices. If the device has an option to limit network access based on a source IP address it is recommended to use this mechanism for increased security.

SNMP versions 3 basics

SNMP version 3 is more complicated because it allows safe authentication and authorization. The authentication mechanism uses the users' database stored on the SNMP agent. Users belong to groups which are used by the authorization mechanism. Groups are used to control which parts of information presented by the SNMP agent

can be accessed. From the network security perspective there are three security models possible in SNMP version 3:

1. no encryption,
2. encrypted authentication credentials but session unencrypted,
3. both authentication credentials and session encrypted.

It is recommended to use security models 2 and 3 with AKK@DA.

SNMP traps

At present AKK@DA does not support SNMP traps.

☝ A full description of the SNMP protocol is not a purpose of this users' guide. For more information, refer to other sources (e.g. "Essential SNMP" by Douglas Mauro, Kevin Schmidt, 2001 , O'Reilly). RFC 1157, RFC 1902, RFC 2571, RFC 2572, RFC 2573, RFC 2574, and RFC 2575 are aslo useful.

6.1.4. Preparing the MS Windows server

At present MS Windows operating systems support only SNMP version 1 and 2c protocols. The SNMP agent configuration is available at: **Start** -> **Control Panel** -> **Administrative tools** -> **Services** -> double click "**SNMP service**" -> tab "**Security**":

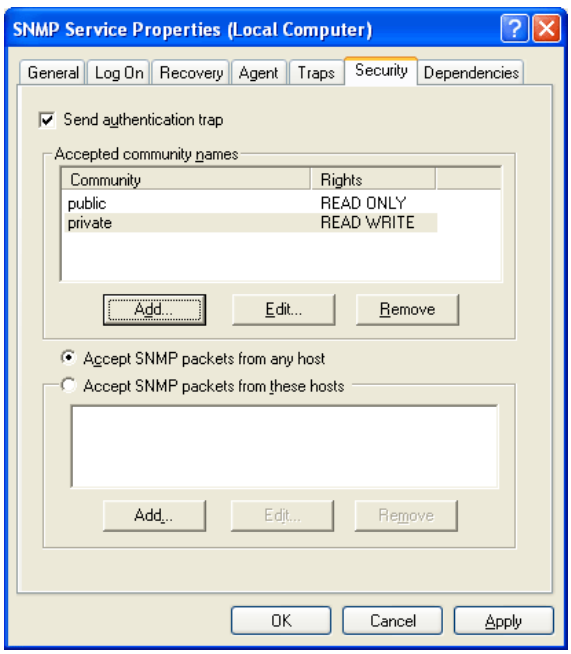


Figure 3. MS Windows SNMP service configuration

From "**Accepted community names**" read-write community should be removed and read-only community should be changed from default "**public**" to your own.

Also "**Accept SNMP packets from these hosts**" should be selected and IP addresses of the AKK@DA's server should be added to the list of permitted IP addresses.

After changes "**SNMP service**" should be restarted.

Installing SNMP service

If "**SNMP service**" is not available on the list of services it needs to be installed. To do this open "**Control Panel**" and start "**Add or Remove Programs**". In the "**Components**" list, check the box "**Management and Monitoring Tools**". Click the "**details**" button and check the "**Simple Network Management Protocol**". "**WMI SNMP Provider**" is not needed. Approve changes and the installation procedure will start. You may be prompted to provide the MS Windows install disk. After installation it is needed to restart the computer.

To test if the SNMP agent configuration is correct, try to access the host SNMP agent with command **snmpwalk** from the AKK@DA's server console (see command `snmpwalk --help` for options). An example of successful output:

```
[root@localhost]# snmpwalk -c secret -v 2c 10.10.1.1 system
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 8 Stepping 10 AT/AT
COMPATIBLE - Software: Windows Version 5.2 (Build 3790 Uniprocessor Free)
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.2
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (302440171) 35 days, 0:06:41.71
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: EXAMPLE-WINDOWS-SRV
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 76
```

6.1.5. Preparing the Linux/UNIX server

This subchapter covers the open source Net-SNMP daemon which is currently the most popular Linux/UNIX SNMP agent (http://www.net-snmp.org/). Only the SNMP protocol version 3 with encrypted authentication and session is described. For other configuration options please refer to the Net-SNMP manual.

To verify if the Net-SNMP SNMP agent is installed run command:

```
[root@localhost]# snmpd -v

NET-SNMP version:  5.4.1
Web:               http://www.net-snmp.org/
Email:             net-snmp-coders@lists.sourceforge.net
```

If this fails you have to install the Net-SNMP daemon. For more information see the Net-SNMP documentation at http://www.net-snmp.org/.

1. Stop the **snmpd** daemon if started: `[root@localhost]# service snmpd stop`

2. Add to the **snmpd.conf** file (usually located in /etc/snmp or /usr/local/etc/ directories, but it depends on specific Linux/UNIX distribution) following line:

```
rouser <username>
```

3. Add to the **/var/net-snmp/snmpd.conf** file the following line (if the file does not exist create it):

```
createUser <username> (MD5|SHA) auth_passwork [DES|AES] priv_passpassword
```

   where:
   -username: is the same as in 2nd step
   -auth_password: password used for authentication
   -priv_password: password used for session encryption
   -auth_password and priv_passowrd may be the same
   -MD5, SHA, DES, AES: all are supported by AKK@DA

4. Start the **snmpd** daemon: `[root@localhost]# service snmpd start`

5. Check the **/var/net-snmp/snmpd.conf** file – line added in step 3 should disappear. Instead of it you should see something similar to:

```
usmUser 1 3 0x80001f8880a2312d038efcdf46 0x617478736e6d7000 0x617478736e6d7000
NULL .1.3.6.1.6.3.10.1.1.2 0xcb4035731b548d65d5e9f3093ad3e2d5
.1.3.6.1.6.3.10.1.2.2 0xcb4035731b548d65d5e9f3093ad3e2d5 0x00
```

To test if the SNMP agent configuration is correct, try to access host SNMP agent with command **snmpwalk** from the AKK@DA's server console (see command `snmpwalk --help` for options). An example of successful output:

```
[root@localhost]# snmpwalk -v 3 -l authPriv -u username -a MD5 -A password -x des -X
password 127.0.0.1 system
SNMPv2-MIB::sysDescr.0 = STRING: Linux sg-atx-mon5 2.6.9-55.ELsmp #1 SMP Fri Apr 20
17:03:35 EDT 2007 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (520775136) 60 days, 6:35:51.36
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: EXAMPLE-LINUX-SRV
SNMPv2-MIB::sysLocation.0 = STRING: somewhere
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (21) 0:00:00.21
```

6.1.6. Preparing the Cisco IOS device

1. Create an access list to protect access to the SNMP agent based on IP addresses:

```
Router(config)#access-list 50 permit ip <AKK@DA's server ip address>
```

2. Create an SNMP group with authentication and session encryption:

```
Router(config)#snmp-server group <group name> v3 priv read v1default access 50
```

3. Create an SNMP user:

```
Router(config)#snmp-server user <user name> <group name> v3 \
auth md5 <authentication password> \
priv des56 <session encryption password> access 50
```

The **priv** option isn't always available to encrypt session, depending on IOS version. If it's not available only **auth** should be used to configure the device SNMP agent using safe authentication at the very least.

To test if the SNMP agent configuration is correct use the same procedure as described in the **Preparing Linux/UNIX server** subchapter.

6.1.7. Preparing the Cisco CatOS device

1. Create an SNMP view:

```
Console> (enable) set snmp view <view name> 1.3.6.1 included
```

2. Createan an SNMP group:

```
Console> (enable) set snmp access security-model v3 privacy read <view name>
```

3. Create an SNMP user:

```
Console> (enable) set snmp user <user name> authentication \
        md5 <authentication password privacy <session encryption password>
```

4. Add the user to the group:

```
Console> (enable) set snmp group <group name> user <user name> security-model v3
```

To test if the SNMP agent configuration is correct use the same procedure as d escribed in the **Preparing Linux/UNIX server** subchapter.

6.1.8. Preparing other devices

In the case of other devices, refer to their manuals to find out how to configure their SNMP agents. Always remember to change the default community read-only string and to remove the read-write string in the case of SNMP v1 and 2c. If possible use the

SNMP version 3. If possible protect access to the SNMP agent with IP address based access control lists.

6.2. Entities essentials

Entity is a general name of any object which represents any monitored device, resource, service or organization unit in AKK@DA. Entities in AKK@DA are organized in a tree structure called **location tree** (GUI -> left panel -> tab **tree** -> folder **locations**).

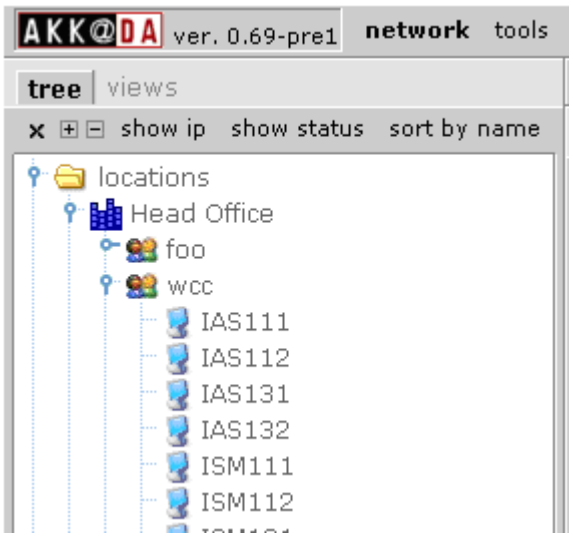


Figure 4. An example location tree

There are 3 types of entities and there are a couple of rules how they can be used:

Name	Description
Group	virtual object; exists to organize **node** entities in the **location tree** structure; an organization unit can contain **node** entities and other **groups** as well only a **group** entity can exist at the root level of the **location tree** has to be created manually
Node	represents any real host (e.g. server, router, switch, etc.) can contain only **service** entities can exist only as a branch of the **group** entity has to be created manually

Service	represents any resource or service of a real host (e.g. CPU, NIC, process, HDD, etc.) cannot contain anything can exist only as a leave of the **node** entity is mostly created automatically; some specific kind s of services have to be created manually

Table 21. Entity types

6.3. Creating groups

1. Log on to AKK@DA as a user with administrative rights.

2. In top menu bar click **network**.

3. In the left panel expand the **locations** folder and right-click the group (or the **locations** folder if you want to add a root level group) to which you want to add a new child group. A context menu will appear. From the context menu select the **add group** menu.

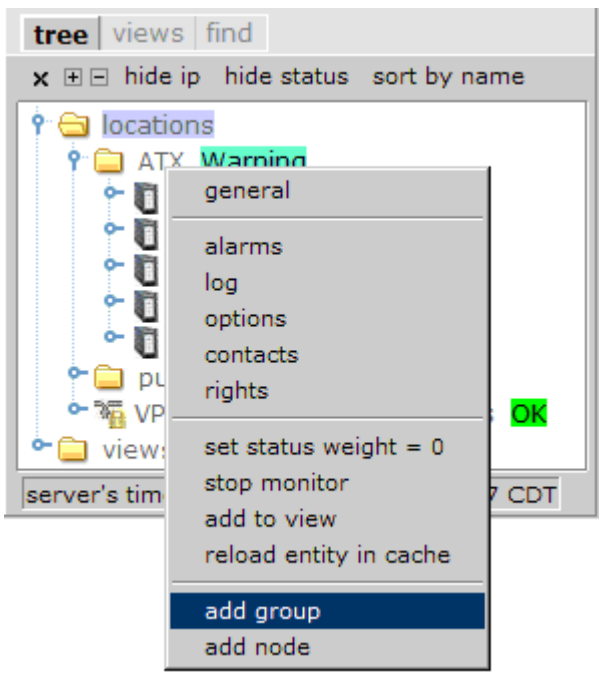


Figure 5. Add group

4. In the right panel the **add group** form will appear. Fill in the form and click the **add group** button.

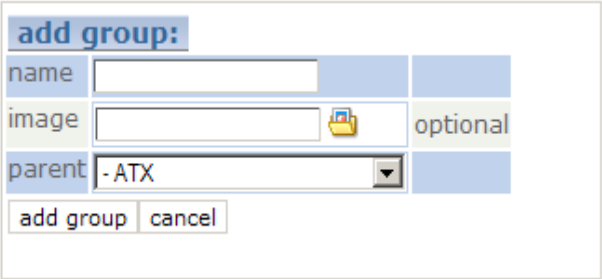


Figure 6. Add group form

Name	Description
name	name of a group
image	optional, click the icon on the right side of the text field to select the icon you want to tie to this group
parent	from the pop-up menu you can select a branch in the location tree you want to add this group

Table 22. Add group form

5. The new group will appear in the chosen branch of the **location tree**.

6.4. Adding a host

6.4.1. Procedure

1. Log on to AKK@DA as a user with administrative rights.
2. In top menu bar click **network**.
3. In the left panel expand the **locations** folder and right-click a group which you want to add a new host to. A context menu will appear. From the context menu select the **add node** menu.

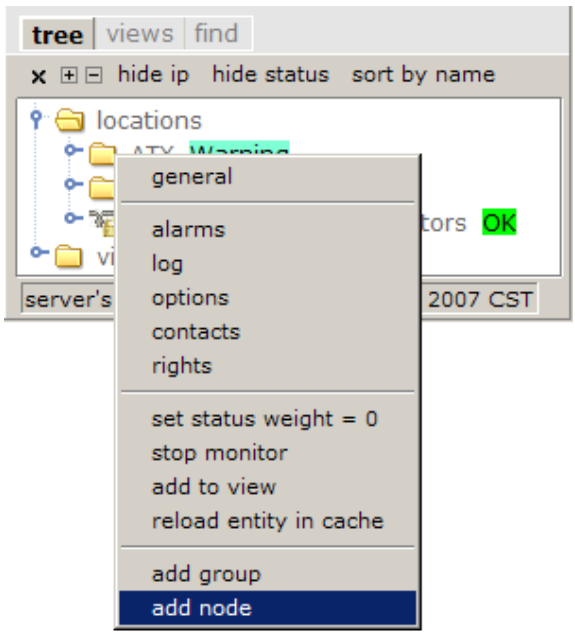


Figure 7. Add node

4. In the right panel the **add node** form will appear. Fill in the form and click the **add node** button.

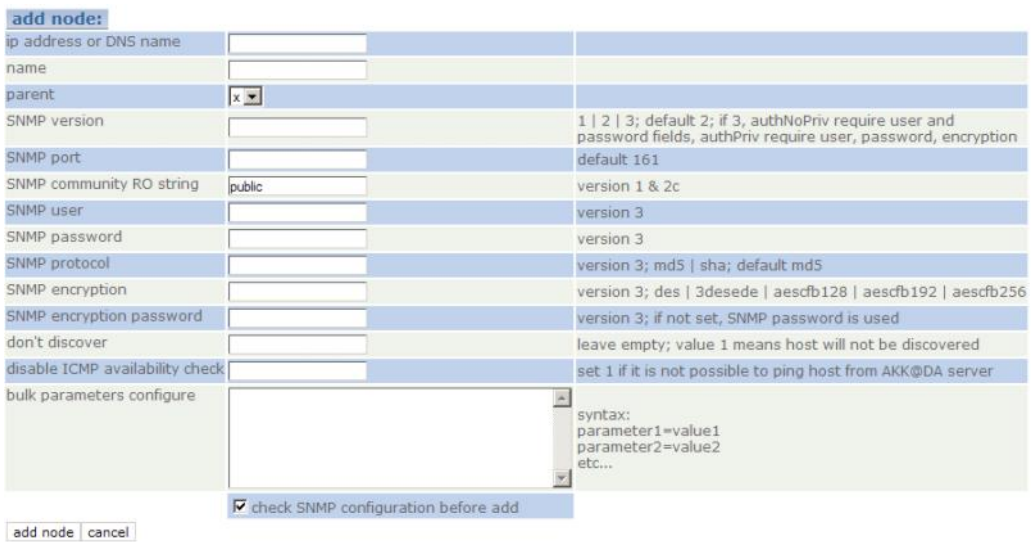


Figure 8. Add node form

Options in bold in the table below are always mandatory and have to be filled in. Other parameters are optional and depend on which SNMP protocol version is used. More details are described in the next subchapters.

Name	Description
ip address or DNS name	mandatory; IP address or DNS name of

	the node
name	optional; will be set automatically to the name reported by node SNMP agent; should be set manually only if the node has no SNMP support
parent	mandatory; from the pop-up menu you can select a branch in the location tree which you want to add this node to
SNMP version	optional; defines the SNMP protocol version; if not set, the SNMP version 2c is used by default
SNMP port	optional; by default port UDP 161 is used; should be defined only when the node SNMP agent works on a different UDP port
SNMP community RO string	optional; read-only community string; default value can be configured in the **/akkada/etc/akkada.conf** file, option **SNMPCommunityRODefault**
SNMP user	optional; an SNMP user used for authentication when using the SNMP version 3 protocol
SNMP password	optional; a password used for authentication when using the SNMP version 3
SNMP protocol	optional; hash algorithm used for authentication when using the SNMP version 3; by default **md5** is used; available options: **md5**, **sha**; values are case sensitive
SNMP encryption	optional; algorithm used for session encryption when using the SNMP version 3; no default; available options: **des**, **3desede**, **aescfb128**, **aescfb192**, **aescfb256**; values are case sensitive
SNMP encryption password	optional; password used for session encryption when using the SNMP version 3;
don't discover	optional; normally should be left empty; if set to 1 a node will not be discovered for available services
disable ICMP availability checks	optional; normally should be left empty; if set to 1 node availability will not be

	checked which may highly decrease AKK@DA performance
bulk parameters configure	optional;
Check SNMP configuration before add	if checked, the host will be checked with the provided SNMP configuration if it's correct; it is suggested to leave it checked; this option should be disabled only when adding a node which does not support the SNMP protocol

Table 23. Add node form

5. A new node will appear in the chosen branch of the **location tree**. After this AKK@DA will start discovering **services** available on the added **node** and automatically adding them to the configuration. The discovery process may take a while. The results should be possible to see in the right panel when you left-click the **node** object in the **location tree**.

6.4.2. Adding an SNMP version 1 node

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Enter **1** into the **SNMP version** field.

Enter a community string to **SNMP community RO string**.

Check the **check SNMP configuration before add** option.

Leave all other fields blank.

6.4.3. Adding an SNMP version 2c node

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Leave **SNMP version** blank.

Enter a community string to **SNMP community RO string**.

Check the **check SNMP configuration before add** option.

Leave all other fields blank.

6.4.4. Adding an SNMP version 3 node

Authentication without session encryption

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Enter **3** into the **SNMP version** field.

Leave the **SNMP community RO string** field blank.

Enter an SNMP user name into the **SNMP user** field.

Enter an SNMP user password into the **SNMP password** field.

Leave **SNMP protocol** blank if you use MD5 as the authentication algorithm or enter **sha** if you use SHA.

Check the **check SNMP configuration before add** option.

Leave all other fields blank.

Authentication with session encryption

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Enter **3** into the **SNMP version** field.

Leave the **SNMP community RO string** field empty.

Enter an SNMP user name into the **SNMP user** field.

Enter an SNMP user password into the **SNMP password** field.

Leave **SNMP protocol** blank if you use MD5 as the authentication algorithm or enter **sha** if you use SHA.

Enter the applicable encryption algorithm name into **SNMP encryption**. For available options, see subchapter **Procedure** in this chapter.

Leave the **SNMP encryption password** field empty if you use the same password for authentication and session encryption otherwise use the correct password for session encryption.

Check the **check SNMP configuration before add** option.

Leave all other fields blank.

6.4.5. Adding a non SNMP node

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Leave all SNMP fields blank.

Enter the node name into the **name** field.

Uncheck the **check SNMP configuration before add** option.

Leave all other fields blank.

6.4.6. Adding a multi SNMP agent node

The number of supported SNMP agents at the same node is unlimited.

For configuring multiple SNMP instances at the node use syntax **foo1::foo2::foo3** for every SNMP related field of the **add node** form (fields whose names start with **SNMP**). String **::** is used as a delimiter.

AKK@DA detects multi SNMP agent configuration based on the **SNMP version** field value

Check the **check SNMP configuration before add** option – AKK@DA will check all configured instances and report possible mistakes before adding a new node.

Example

Let's say an example node has 3 instances of SNMP agents:

1. version 3, authentication without encryption, user name JOHN, user password 12345678, authentication algorithm SHA, port 161
2. version 2c, community string PUBLIC, port 1616
3. version 3, authentication with encryption, user name BOB, user password ABCDEF, authentication algorithm MD5, encryption algorithm DES, encryption password QAZQAZ, port 2000

See the configuration below, pay careful attention to the number of delimiters **::**.

name	Value

SNMP version	3**::::**3
SNMP port	**::**1616**::**2000
SNMP community RO string	**::**PUBLIC**::**
SNMP user	JOHN**::::**BOB
SNMP password	12345678**::::**ABCDEF
SNMP protocol	sha**::::**
SNMP encryption	**::::**des
SNMP encryption password	**::::**QAZQAZ

Table 24. An example configuration of node with multi SNMP agents

AKK@DA splits each value listed above based on the **::** delimiter. (e.g. **::::des** means that instance **1** value is **empty**, instance **2** value is **empty**, instance **3** value is **des**; **::PUBLIC::** means that instance **1** value is **empty**, instance **2** value is **PUBLIC**, instance **3** value is **empty**). Any empty value is interpreted in the same way as described in subchapter **Procedure** of this chapter.

6.4.7. Adding a node monitored only by ICMP protocol

Enter the correct IP address or DNS name into the **ip address or DNS name** field.

Leave all SNMP fields blank.

Enter the node name into the **name** field.

Enter **1** into the **don't discover** field.

Uncheck the **check SNMP configuration before add** option.

Leave all other fields blank.

Right-click on the newly added node object in the **location tree**.

Select the **add service** from the context menu.

Add the ICMP service (see chapter **Services**, subchapter **Adding the service manually**).

6.4.8. Adding a node which is not accessible via ICMP

Configure the node in the same way as described in previous subchapters (depending on your needs) and enter **1** into the **disable ICMP availability checks** field.

Chapter 7

7. Services

7.1. Discovery process

General

Generaly most service types are discovered and configured automatically by AKK@DA discovery process **nm-discover.pl** (for more information regarding specific service types, refer to the **Probes** subchapter in this chapter). The discovery process uses logic defined in probe modules for finding and configuring supported services. It does not have any detection logic itself. The **nm-discover.pl** module is only responsible for managing discovery process and performs discovery process using the probe logic.

Generally services which are discovered automatically cannot be added manually. This rule does not cover **tcp_generic** and **ssl_generic** services which are discovered automatically but only in the case of well known ports 1-1024. These kinds of services have to be configured manually for ports other than 1-1024 (see the **Adding the service manually** subchapter in this chapter).

The discovery process is performed automatically after a new node has been added and then every 10 hours.

Types of discovery

In the case of specific types of probes the discovery process can work in a couple of ways..

Type	Description
Automatic	The most common type of discovering. No special action is required, services are discovered without any additional administrator action.
No discover	There are no discovery procedures available and services with that

	type of discovering have to be configured manually
Mixed	When the node is configured services with mixed type of discovering are not discovered. The administrator has to start discovering manually (see subchapter **Starting the discovery process** in this chapter). If at least one service with that type of discovering will be discovered after it was manually started, AKK@DA will start to discover these types of services on this particular host automatically. This type of discovering is normally used when some additional node configuration is required to discover these kinds of services (e.g. see the **Probe** subchapter**,** paragraph **softax_ima** in this chapter).
Manual	Discovery procedures are available but discovering has to be started manually when needed.

Table 25. Types of discovering

Starting the discovery process

It is also possible to force the discovery process manually when needed by using the following procedure:

1. In the left panel right-click on an applicable **node** object in the **location tree**.
2. Choose the **options** menu from the context menu.
3. In the right panel click the **discover** button. The d**iscover** form will appear.

Table 26. Discover form

4. Select probe type to discover specific type services or leave **all probes** to discover all supported services and then click **process**.

The discovery process can take a few minutes.

7.2. Adding a service manually

There is a group of services which are not discovered automatically because of their nature. In the case of these services manual adding is required. To add a service manually, please follow the procedure below.

1. In the left panel right-click on an applicable **node** object in the **location tree**.
2. Choose the **add service** menu from the context menu. In the right panel the **add service** form will appear.
3. From the pop-up menu of the **add service** form select the type of the service you need. The page will reload and there will be more options in the **add service** form available depending on the selected type of service.
4. Enter the name of the service into the **name** field.
5. Enter applicable values into the rest of the fields (to get more help regarding the function of these fields, refer to the description of the selected type of service in the **Probes** subchapter in this chapter)
6. Click the **add service** button. The new service will appear in the right panel under the selected node object.

7.3. Probes

7.3.1. General rules of using parameters and configuration files

All parameters defined in AKK@DA are inherited one level down based on the **location tree** structure. This means, any parameter configured on the host entity is inherited to all services of this host. Any parameter configured on the group entity is inherited to all its group and node entities, but not to the services of these hosts and not to the subgroups of these groups.

The functions of all parameters are described later in this chapter.

Any optional parameter can be configured via the web GUI in the **right panel**, the **options** tab of the selected entity. There are 3 forms available.

Updating and deleting

The first one is for updating and deleting parameters:

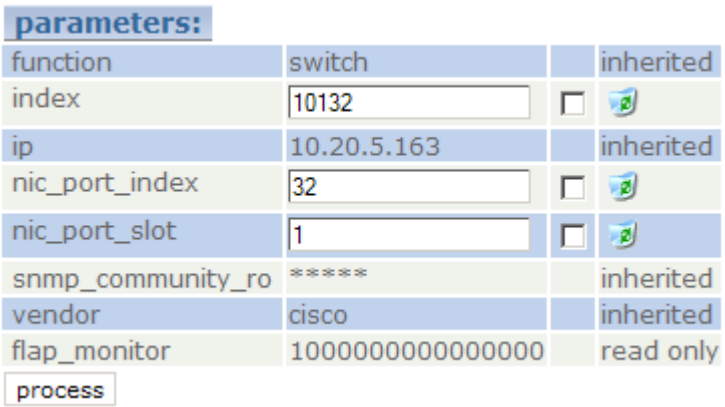


Figure 9. Updating parameters

Inherited parameters are read-only. Use check boxes to delete specific parameters. After the changes have been made, click the **process** button.

Adding

The second form is for adding parameters:

Figure 10. Adding parameters

Select the applicable parameter name from the pop-up menu, enter the correct value of this parameter into the text field and click the **process** button to add the selected parameter.

Bulk modifying

The last form is for modifying bulk parameters. This is a more difficult way to modify parameters, but it's time effective when modification of many entities is needed.

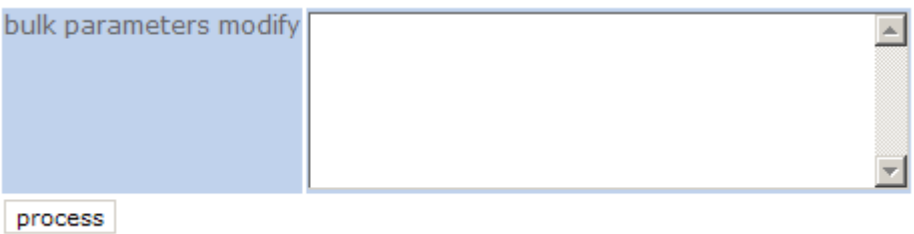


Figure 11. Modifying bulk parameters

Enter the correct modifications into the **bulk parameters modify** field and click the **process** button. Any syntax errors will be reported. The expected syntax is simple. To add parameters or modify their values use:

```
parameter1=value1
parameter2=value2
...
```

If deleting is needed use:

```
parameter1=%%DELETE%%
```

%%DELETE%% key word is case sensitive. Adding, modifying and deleting can be mixed.

E.g. to add a new host with the IP address 1.1.1.1, SNMP version 2c, community string public use:

```
ip=1.1.1.1
snmp_community_ro=public
snmp_version=2
```

Configuration files

Some probes have their own configuration files which are described individually in the specific subchapters in this chapter. After any changes in these files have been made the affected probe and the **nm-discover.pl** process must be restarted otherwise changes will not work.

7.3.2. Common parameters

Parameters listed in the table below are generally used by all types of probes. Parameters listed in the following subchapters regarding specific probes are interpreted only by specific probes.

Name	Type	Description
attempts_max_count	Optional	Number of attempts returning the **not OK** status required for an alarm to be raised
attempts_retry_interval	Optional	Retry interval in seconds (default **30**) which is used for subsequent attempts when the **not OK** status appears; used when **attempts_max_count** is defined
availability_check_disable	Optional	1 = AKK@DA does not check the host availability with the ICMP protocol; **may highly decrease AKK@DA performance**
disable_error_message_change_log	Optional	Overwrites the default value defined in the **/akkada/etc/Akkada.conf** file **DisableErrorMessageLog**; 1 = when the alarm is raised and

		an error message has changed, this change will not be logged in the history table; sometimes it's needed to avoid a storm of inserts into the history table
dont_discover	Optional	1 = means AKK@DA will not discover any services available on the specific host without the discovery process being forced manually
flaps_alarm_count	Optional	Overwrites the default value defined in the **/akkada/etc/Akkada.conf** file **FlapAlarmCount**; the number of status changes after which the **flap** status will be set; should be a value between **2-8**
flaps_disable_monitor	Optional	1 = disables flap detection
function	Optional	Defines the function icon of the entity used in the web GUI; however group entities shouldn't be modified manually
ip	Mandatory Inherited from host	IP address
oids_disabled	Optional	The list of unsupported SNMP OIDS on the specific host. Set automatically, shouldn't be modified manually.
snmp_authpassword	Mandatory (SNMP v3) Inherited from host	SNMP user's authentication password
snmp_authprotocol	Mandatory (SNMP v3) Inherited from host	SNMP user's authentication protocol. Default **md5**. Available options: **md5**, **sha**.
snmp_community_ro	Mandatory (SNMP v1&2c) Inherited from host	SNMP read-only community string
snmp_privpassword	Optional (SNMP v3) Inherited from host	SNMP encryption password
snmp_privprotocol	Optional	SNMP encryption protocol.

	(SNMP v3) Inherited from host	Available options: **des**, **3desede**, **aescfb128**, **aescfb192**, **aescfb256**.
snmp_retry	Optional Inherited from host	SNMP request retry count. Default **1**.
snmp_timeout	Optional Inherited from host	SNMP request timeout (in seconds). Default **5** sec.
snmp_user	Mandatory (SNMP v3) Inherited from host	SNMP user's name
snmp_version	Optional Inherited from host	SNMP version. Default **2**. Available options: **1** (version 1), **2** (version 2c), **3** (version 3).
stop_discover	Optional	List of probe names which shouldn't be discovered on the specific host. Format: **Name1::Name2** …
vendor	Optional	Defines the vendor icon of the entity used in the web GUI; shouldn't be modified manually; is set automatically if a/the node vendor is detected and supported

Table 27. Common parameters

7.3.3. bgp_peer

Overview

The probe monitors BGP peer sessions and raises an alarm if it detects errors in a BGP session or a BGP session state is other than active.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
bgp_peer_errors_ignore	Optional	1 = don't alarm BGP session errors
bgp_peer_state_ignore	Optional	1 = don't alarm BGP session bad states

Table 28. bgp_peer parameters

7.3.4. cisco_css_content

Overview

The probe monitors Cisco CSS 11000 series device content rules. It raises an alarm when content rule remains in a bad state, all content rule services are down or utilization thresholds are exceeded.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
cisco_css_content_index_1	Mandatory	Content rule owner index; **do not change it**
cisco_css_content_index_2	Mandatory	Content rule index; **do not change it**
cisco_css_content_stop_warning_suspended_state	Optional	1 = don't alarm when a/the rule is suspended
threshold_high	Optional	high utilization threshold (%); overwrites default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdHighDefault**
threshold_medium	Optional	medium utilization threshold (%); overwrites default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdMediumDefault**

Table 29. cisco_css_content parameters

Set parameters

cisco_css_content_index_1, **cisco_css_content_index_2**

7.3.5. cisco_css_owner

Overview

The probe monitors Cisco CSS 11000 series device owners. It raises an alarm when previously existing owner information is unavailable. The general purpose of this probe is to collect owner statistics.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
index	Mandatory	Owner index; **do not change it**

Table 30. cisco_css_owner parameters

Set parameters **index**

7.3.6. cisco_css_service

Overview The probe monitors Cisco CSS 11000 series device services. It raises an alarm when: the service stays in a bad state, service's keepalive stays in a bad state or utilization thresholds are exceeded.

Discover Automatic

Test method SNMP

Used parameters

Name	Type	Description
cisco_css_service_stop_warning_down_state	Optional	1 = don't alarm when the service is down
cisco_css_service_stop_warning_high_average_load	Optional	1 = don't alarm when the service average load is high
cisco_css_service_stop_warning_high_load	Optional	1 = don't alarm when any type of service load is high
cisco_css_service_stop_warning_high_long_load	Optional	1 = don't alarm when the service long load is high
cisco_css_service_stop_warning_high_short_load	Optional	1 = don't alarm when the service short load is high
cisco_css_service_stop_warning_suspended_state	Optional	1 = don't alarm when the service is suspended
index	Mandatory	Service index; **do not change it**
threshold_high	Optional	High utilization threshold (%); overwrites default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdHighDefault**
threshold_medium	Optional	Medium utilization threshold (%); overwrites default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdMediumDefault**

Table 31. cisco_css_service parameters

Set parameters **index**

7.3.7. cisco_dial_peer_voice

Overview

The probe monitors Cisco device voice dial peers. It raises an alarm when previously existing dial peer information is unavailable. The general purpose of this probe is to collect owner statistics.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
ianaiftype	Mandatory	IANA ifType MIB definition; **do not change it**
index	Mandatory	Dial peer index; **do not change it**

Table 32. cisco_dial_peer_voice parameters

Set parameters

index, ianaiftype

7.3.8. cisco_pix_ipsec

Overview

The probe connects to Cisco PIX firewall, obtains failover status and IKAKMP SA of the current sessions status. The probe raises an alarm if there is any SA session in other than QM_IDLE state. It also reports an error if the expected SA session does not exist or when the expected SA session exists on the standby firewall.

Discover

No discover. Service has to be defined manually.

Test method

SSH

Used parameters

Name	Type	Description
cisco_pix_ipsec_alarm_down	Optional	List of IP addresses of peers which are expected to always be up, otherwise the alarm should be raised; format: **ip1||ip2||ip3**…
cisco_pix_ipsec_dont_alarm_wrong_state	Optional	List of IP addresses of peers which are to be ignored for SA session statuses (disable QM_IDLE checking); format: **ip1||ip2||ip3**…
cisco_pix_ipsec_enable	Mandatory	Password to switch to the privileged mode on PIX; privileged mode must have commands "**show failover**" and "**show crypto isakmp sa**" available
cisco_pix_ipsec_names	Optional	If you'd like to see peer names

		instead of IP addresses; format: **ip1::name1||ip2::namne2**…
cisco_pix_ipsec_password	Mandatory	PIX firewall user password
cisco_pix_ipsec_username	Mandatory	PIX firewall user name

Table 33. cisco_pix_ipsec parameters

Issues

The **akkada** user on the AKK@DA's server must have read-write rights to the **root's ~/.ssh** directory and to the **~/.ssh/known_hosts** file.

7.3.9. cpu

Overview

The probe monitors CPU. It supports standard hosts MIB, UCD MIB (standard Linux SNMP agent), Altiga, Arrowpoint, Cisco and HP devices. The probe raises an alarm when the CPU utilization exceeds defined thresholds. In the case of UCD alarms reported by the UCD SNMP agent are also raised

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
cpu_count	Optional	Number of CPU reported by the hosts MIB; set automatically; **do not change it manually**
cpu_host_resources_utilization_aggregate	Optional	1 = utilization alarms are raised based on the aggregated CPU utilization when more than one CPU is available (by default alarms are raised per specific CPU); this option is used if CPU is reported by **hosts** MIB
cpu_stop_warning_high_utilization	Optional	1 = don't raise alarm when defined thresholds are exceeded
cpu_type	Mandatory	Defines detected CPU type; set automatically; **do not change it manually**
cpu_ucd_la_15_threshhold	Optional	Load average 15 minutes threshold; if not defined, threshold is not checked; Default alarms reported by the SNMP agent are still enabled; this option is used if CPU is reported by UCD MIB
cpu_ucd_la_1_threshhold	Optional	Load average 1 minute threshold; if not defined, threshold is not checked; Default alarms reported by the SNMP agent are still enabled; this option is

		used if the CPU is reported by UCD MIB
cpu_ucd_la_5_threshhold	Optional	Load average 5 minutes threshold; if not defined, threshold is not checked; Default alarms reported by the SNMP agent are still enabled; this option is used if CPU is reported by UCD MIB
cpu_ucd_utilization_aggregate	Optional	1 = utilization alarms are raised based on the aggregated CPU utilization (by default alarms are raised per specific type of CPU time consumer, e.g. kernel, user, wait, etc.); this option is used if CPU is reported by UCD MIB
threshold_high	Optional	High utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdHighDefault** is used
threshold_medium	Optional	Medium utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdMediumDefault** is used

Table 34. cpu parameters

Set parameters

cpu_count, cpu_type

Configuration

Default configuration is defined in the **/akkada/etc/conf.d/Probes/cpu.conf** file.

```
{
    'ucd' => {
        fillIdleOnGraphsPercent => 1,
        fillIdleOnGraphsRaw => 0,
    },
}
```

Name	Description
fillIdleOnGraphsPercent	1 = idle track is shown on the percentage utilization graph, otherwise it isn't
fillIdleOnGraphsRaw	1 = idle track is shown on the raw utilization graph, otherwise it isn't

Table 35. cpu configuration parameters

7.3.10. dns_query

Overview

The probe sends a defined DNS query to the DNS server and checks the server

answer. It raises an alarm in the case of any error and when the server answer doesn't match the configured expected value. It also reports an error if any defined threshold is exceeded.

Discover

No discover

Test method

DNS

Used parameters

Name	Type	Description
dns_query_expected_value	Mandatory	Expected DNS server answer
dns_query_field	Mandatory	Name of the field (see man Net::DNS::RR::) where **dns_query_expected_value** is looked for; e.g. for **dns_query_record_type** A it can be "address" (see man Net::DNS::RR:A), for **dns_query_record_type** MX it can be "exchange" or "preference" (see man Net::DNS::RR::MX)
dns_query_query	Mandatory	Query value; e.g. www.sample.pl, 192.168.1.1, 1.1.168.192.in-addr.arpa., etc.
dns_query_record_type	Mandatory	Query record type; e.g. A, CNAME, MX, etc. (see man Net::DNS::RR)
threshold_high	Optional	High threshold for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used
threshold_medium	Optional	Medium threshold for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used
timeout	Optional	Timeout for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used

Table 36. dns_query parameters

Configuration

The default configuration is defined in the **/akkada/etc/conf.d/Probes/dns_query.conf** file.

```
{
      'DefaultTimeout' => 1,
      'ThresholdHighDefault' => '0.9',
      'ThresholdMediumDefault' => '0.5'
}
```

Name	Description
DefaultTimeout	Default timeout for the DNS server answer time

	(seconds)
ThresholdHighDefault	Default high threshold for the DNS server answer time (seconds)
ThresholdMediumDefault	Default medium threshold for the DNS server answer time (seconds)

Table 37. dns_query configuration parameters

7.3.11. dns_server

Overview

The probe checks if the DNS server is running and answering. It raises an alarm in the case of any error other than NOERROR or NXDOMAIN (see man Net::DNS::Resolver). It also reports an error if any defined threshold is exceeded.

Discover

Automatic

Test method

DNS

Used parameters

Name	Type	Description
dns_server_hostname	Optional	Host name used for checking if the DNS server is alive; doesn't have to be an existing host name; if not defined the default value **DiscoverHostNameDefault** from the probe configuration is used
threshold_high	Optional	High threshold for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used
threshold_medium	Optional	Medium threshold for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used
Timeout	Optional	Timeout for the DNS server answer time (seconds); if not defined the default value from the probe configuration is used

Table 38. dns_server parameters

Configuration

The default configuration is defined in the **/akkada/etc/conf.d/Probes/dns_server.conf** file.

```
{
      'DefaultTimeout' => 1,
      'DiscoverHostNameDefault' => '127.0.0.1',
      'ThresholdHighDefault' => '0.9',
      'ThresholdMediumDefault' => '0.5'
}
```

Name	Description
DefaultTimeout	Default timeout for the DNS server answer time

	(seconds)
DiscoverHostNameDefault	Default host name used for discovering the DNS servers and for checking if the DNS server is alive; doesn't have to be an existing host name
ThresholdHighDefault	Default high threshold for the DNS server answer time (seconds)
ThresholdMediumDefault	Default medium threshold for the DNS server answer time (seconds)

Table 39. dns_server configuration parameters

7.3.12. group

Overview

By default this probe doesn't monitor anything and its purpose is to group other group and host entities into the **location tree**.

If there is a defined parameter **ip** for a group type entity, the availability of this IP address is monitored with ICMP by the **nm-available.pl** process. When this IP address is unavailable, the status of the group is set to UNREACHABLE and the status of all of the children entities in this group are set to UNKNOWN. This is a mechanizm that reduces the number of UNREACHABLE alarms.

E.g. if there is a remote location, all of the hosts located in this locatrion should be organized in one group (with subgroups if needed); the gateway IP address of this location (from AKK@DA's perspective) should be configured at this group; when the gateway is unreachable, only one alarm will be raised to show that this groups is unreachable instead of raising a lot of unreachable alarms for each host separately.

Discover

No discover

Test method

ICMP (optionally)

Configuration

The default configuration is defined in the **/akkada/etc/conf.d/Probes/group.conf** file.

```
{
      'not_tested' => 1,
}
```

Name	Description
not_tested	Shouldn't be modified; this is an internal parameter

Table 40. group configuration parameters

7.3.13. hdd

Overview

The probe monitors hard drives. It supports standard **hosts** MIB and UCD MIB (standard Linux SNMP agent). In the case of **hosts** MIB it monitors RAM and virtual memory as well because hosts MIB reports them as a kind of storage. The probe raises an alarm when disk usage exceeds defined thresholds.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
hdd_stop_raise_inode_alarms	Optional	1 = ignore inode utilization; used if hard drive is reported by UCD MIB
hdd_threshold_bytes_mode	Optional	1 = raises an alarm based on the number of free bytes instead of percentage utilization
hdd_threshold_minimum_bytes	Optional	Minimum number of expected free bytes; when the number of free bytes is less than this value, an alarm is raised; used only if **hdd_threshold_bytes_mode** is set to 1
hdd_type	Mandatory	Defines detected disk drive type; set automatically; **do not change it manually**
threshold_high	Optional	High utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdHighDefault** is used
threshold_medium	Optional	Medium utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdMediumDefault** is used

Table 41. hdd parameters

Set parameters

hdd_type

Configuration

The default configuration is defined in the **/akkada/etc/conf.d/Probes/hdd.conf** file.

```
{
      'host_resources' => {
        'Monitor' => {
          'floppy disk' => 0,
          'other' => 0,
          'unknown' => 0,
```

```
          'compact disc' => 0,
          'removable disk' => 0,
          'flash memory' => 0
        }
      }
},
```

Name	Description
Monitor	Used if hard drive is reported by hosts MIB; list of types of storages; if storage type value is set to 0, storage of this type will NOT be discovered during the discovery process; if storage type value is set to 1 or storage type value is not defined, storage of this type will be discovered during discovery process

Table 42. hdd configuration parameters

7.3.14. host_resources_process

Overview

The probe monitors any system process based on information shown by SNMP, host resources MIB (most of the operating systems support this MIB). The probe raises an alarm when the defined process is not running, configured thresholds are exceeded or one or more proccess states are **invalid**.

Discover

No discover. Service has to be defined manually.

Test method

SNMP

Used parameters

Name	Type	Description
host_resources_process_cpu_time_max	Optional	Maximum CPU usage threshold (seconds); it's computed as a summary CPU usage of all processes of a given name; if not defined or set to 0, this condition is not checked
host_resources_process_memory_max	Optional	Maximum memory usage threshold (bytes); it's computed as a summary memory usage by all processes of a given name; if not defined or set to 0, this condition is not checked
host_resources_process_ignore_invalid_state	Optional	1 = ignore invalid state of processes
host_resources_process_max	Optional	maximum expected process count; if not defined or set to 0, this condition is not checked
host_resources_process_min	Optional	minimum expected process count; if not defined or set to 0, this condition is not checked
host_resources_process_path_mode	Optional	1 = AKK@DA looks for process by

		path returned via OID **hrSWRunName** rather than via OID **hrSWRunName**; this is useful when the SNMP agent returns empty **hrSWRunName** values (e.g. OpenVMS) or when certainty that the application is started in the proper path is needed
name	Mandatory	Full name of the process; e.g. sqlagent.exe (see **task manager** on MS Windows systems, **ps** command on UNIX systems); the **ps** command on some UNIX and other operating systems reports different process names than the SNMP protocol – in that case names reported by SNMP (OID **hrSWRunName**) should be used

Table 43. host_resources_process parameters

7.3.15. host_resources_system

Overview

The probe monitors the number of current users and processes based on information shown by SNMP, host resources MIB (most of the operating systems support this MIB). The probe raises an alarm if configured thresholds are exceeded.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
threshold_high	Optional	High threshold used for either the number of current users or processes (number)
threshold_medium	Optional	Medium threshold used for either the number of current users or processes (number)

Table 44. host_resources_system parameters

7.3.16. icmp_monitor

Overview

The probe monitors a specified IP address. It raises an alarm if there are lost packets or defined thresholds are exceeded.

Normally IP addresses are monitored with ICMP automatically (IP addresses of node entities and IP addresses associated with monitored network interfaces, which are reported by SNMP). In both situations ICMP statistics are displayed with node or NIC statistics. This probe allows monitoring IP addresses which are not associated with NIC or nodes.

Discover

No discover

Test method

ICMP

Used parameters

Name	Type	Description
nic_ip	Mandatory	IP address to be monitored
nic_ip_icmp_check_disable	Optional	1 = ICMP test is disabled
nic_ip_icmp_check_max_delay_threshold	Optional	maximum delay threshold (seconds); if not defined, the default value defined in the `/akkada/etc/conf.d/ICMPMonitor.conf` file, **DefaultDelayThreshold** is used
nic_ip_icmp_check_lost_threshold	Optional	maximum lost packet threshold (number 0-20); if not defined, the default value defined in the `/akkada/etc/conf.d/ICMPMonitor.conf` file, **DefaultLostThreshold** is used

Table 45. icmp_monitor parameters

Set parameters

nic_ip_icmp_check_disable

7.3.17. nic

Overview

The probe monitors network interfaces. If an IP address is configured on the network interface, the probe also performs ICMP tests. The probe raises an alarm when an operation status is other than **up** (but only when the administrative status is **up**), it detects error packets, defined thresholds are exceeded, Cisco switches report port duplex disagree state, ICMP packets are lost or ICMP response times exceed defined thresholds.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
index	Mandatory	Interface index; set automatically; **do**

		not change it manually
ip_addresses	Optional	IP addresses related to the network interface; set automatically; **do not change it manually**
nic_ambiguous_ifDescr	Optional	Defined when the host interface names are ambiguous; set automatically; **do not change it manually**
nic_bandwidth	Optional	Interface speed (bits per second); overwrites interface speed reported by SNMP during the calculating of utilization statistics; useful when interface speed reported by SNMP is wrong
nic_bandwidth_aggregate	Optional	1 = inbound and outbound traffic are summarized before being compared with thresholds; by default (without this parameter) AKK@DA compares inbound and outbound traffic with thresholds separately
nic_errors_ignore	Optional	1 = alarms are not raised in the case of error packets
nic_ifOperStatus_ignore	Optional	1 = alarms are not raised in the case of improper operational status
nic_ifOperStatus_interpret_dormant_down	Optional	1 = alarm is not raised when the operational status is **dormant**;
nic_ifOperStatus_invert	Optional	1 = alarm is raised only when the operational status is **up**
nic_ifOperStatus_invert_msg	Optional	Additional text message added to standard error message in the case of an alarm when the **nic_ifOperStatus_invert** parameter value is 1
nic_ip	Optional	IP address associated with the specific network interface; used for ICMP tests; only primary IP address is used if there are more than one IP addresses associated with the network interface; set automatically; **do not change it manually**
nic_ip_icmp_check_disable	Optional	1 = ICMP test is disabled
nic_ip_icmp_check_max_delay_threshold	Optional	maximum delay threshold (seconds); if not defined, the default value defined in the `/akkada/etc/conf.d/ICMPMonitor.conf` file, **DefaultDelayThreshold** is used
nic_ip_icmp_check_lost_threshold	Optional	maximum lost packets threshold (number 0-20); if not defined, the

		default value defined in `/akkada/etc/conf.d/ICMPMonitor.conf` file, **DefaultLostThreshold** is used
nic_port_index	Optional	Cisco switch interface port index; set automatically; **do not change it manually**
nic_port_slot	Optional	Cisco switch interface slot index; set automatically; **do not change it manually**
nic_speed_check_disable	Optional	1 = in the case of exceeding utilization thresholds, alarms are not raised
threshold_high	Optional	high utilization threshold (%); overwrites the default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdHighDefault**
threshold_medium	Optional	medium utilization threshold (%); overwrites the default configured in the **/akkada/conf.d/Probe.conf** file, **ThresholdMediumDefault**

Table 46. nic parameters

Set parameters

index, ip_addresses, nic_ambiguous_ifDescr, nic_ip, nic_port_index, nic_port_slot

Configuration

The default configuration is defined in the **/akkada/etc/conf.d/Probes/nic.conf** file.

```
{
      'DiscoverDisableOperStatusCheckOnDownInterfaces' => 1,
}
```

Name	Description
DiscoverDisableOperStatusCheckOnDownInterfaces	Value 1 means parameter **nic_ifOperStatus_ignore** is set automatically when the network interface operational status during discovery process is not **up**; value 0 disables this mechanism

Table 47. nic configuration parameters

7.3.18. node

Overview

The probe collects basic host information such as system name, description, uptime, etc. It tests SNMP availability and sets the NoSNMP status/alarm (which is inherited to all SNMP based services of the specific host) when SNMP is not available (to disable SNMP testing don't define SNMP parameters on the host en tity). This probe is

also responsible for detecting host function and vendor and setting proper function and vendor icons.

Discover No discover

Test method SNMP

Set parameters **function, ip_forwarding, vendor**

Configuration The probe configuration is kept in the **/akkada/etc/conf.d/Probes/node.conf** file. There are configured definitions used during detecting functions and vendors of hosts.

7.3.19. ntp

Overview The probe monitors NTP servers. It raises an alarm when the NTP server is not accessible and when time synchronization is lost.

Discover Automatic

Test method NTP

Configuration The probe configuration is kept in the **/akkada/etc/conf.d/Probes/ntp.conf** file.

```
{
    ntpq => '/usr/local/bin/ntpq',
    ntpq_params => '-np',
},
```

Name	Description
Ntpq	Full path to the **ntpq** binary
ntpq_params	Parameters for the **ntpq** binary (see **man ntpq**)

Table 48. ntp configuration parameters

7.3.20. ram

Overview The probe monitors the RAM memory. It supports UCD MIB (standard Linux SNMP agent), Arrowpoint, Cisco and HP devices. The probe raises an alarm when RAM usage exceeds defined thresholds.

Discover Automatic

Test method SNMP

Used parameters

Name	Type	Description
ram_disable_memory_full_alarm_real	Optional	1 = don't raise alarm the **real** memory usage exceeds defined thresholds; used if RAM is reported by UCD MIB
ram_disable_memory_full_alarm_swap	Optional	1 = don't raise alarm the **swap** memory usage exceeds defined thresholds; used if RAM is reported by UCD MIB
ram_disable_memory_full_alarm_total	Optional	1 = don't raise alarm the **real+sawp** memory usage exceeds defined thresholds; used if RAM is reported by UCD MIB
ram_threshold_bytes_mode	Optional	1 = raise alarm based on the number of free bytes instead of percentage utilization
ram_threshold_minimum_bytes	Optional	Minimum number of expected free bytes; when the number of free bytes is lower than this value, an alarm is raised; used only if **ram_threshold_bytes_mode** is set to 1
ram_type	Mandatory	Defines detected RAM type; set automatically; **do not change it manually**
threshold_high	Optional	High utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdHighDefault** is used
threshold_medium	Optional	Medium utilization threshold (%); if not defined the default value defined in the **/akkada/etc/conf.d/Probe.conf** file, **ThresholdMediumDefault** is used

Table 49. hdd parameters

Set parameters

ram_type

7.3.21. route

Overview

The probe monitors specific route entries. It raises an alarm when the monitored path is not available and when the next hop value doesn't match the expected value.

Discover

No discover

Test method

SNMP

Used parameters

Name	Type	Description
name	Mandatory	Destination network address (e.g. 0.0.0.0 for default route, 192.168.1.0 for 192.168.1.0 network, etc.); there is no option to define network mask
route_next_hop	Mandatory	Expected IP address of the next hop (e.g. 192.168.1.1)

Table 50 route parameters

7.3.22. snmp_generic

Overview

Generic SNMP probe which is based on the template model. Templates are described in the **Templates for snmp_generic probe** chapter. Generally templates are text files which define how to discover, monitor and present in the web GUI any information which is available through the SNMP protocol.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
snmp_generic_definition_name	Mandatory	Snmp_generic template name; set automatically; **do not change it manually**
snmp_generic_text_test_disable	Optional	1 = the probe will not perform text tests defined by the specific template

Table 51 snmp_generic parameters

Configuration

The probe doesn't have a configuration file in the sense of the configuration file used in the case of other probes. All settings are defined by templates which are stored in the **/akkada/etc/snmp_generic** directory. For template description, see chapter **Templates for snmp_generic probe**.

7.3.23. softax_ima

Overview

The probe monitors Softax ICC components using the Softax SNMP agent. It raises an alarm when detects a component restart, defined minimum/maximum process counts are exceeded, the component information is not available or the component reports error.

Discover

Mixed

Test method

SNMP

Used parameters

Name	Type	Description
index	Mandatory	ICC component index; set automatically; **do not change it manually**
softax_ima_community	Mandatory	Softax SNMP agent read only community string
softax_ima_max_active	Optional	Maximum expected count of ICC component processes; if not set or set to 0 this threshold is not checked; default 0
softax_ima_min_active	Optional	Minimum expected count of ICC component processes; if not set or set to 0 this threshold is not checked; default 1
softax_ima_port	Mandatory	Softax SNMP agent UDP port number; when multiple instances of the Softax SNMP agent exist at the same host, use syntax: **port1::port2::port3**
timeout	Optional	Timeout for Softax SNMP agent queries (seconds); default 5

Table 52 route parameters

Set parameters

index

Configuration

The probe configuration is kept in the **/akkada/etc/conf.d/Probes/softax_ima.conf** file.

```
{
      IgnoreSource => [
          'isg_ptc_http_client.cxx',
          'recharge_application.py',
          'pull_mgr_process.cxx:1293',
          'smsc_stk_push_proc.cxx:69',
          'zsi_gtw_handler.cxx:310',
          'ivr_pko_worker.cxx:2711',
          'zsi_gtw_handler.cxx:352',
          'zsi_gtw_handler.cxx:321',
          'zsi_gtw_contact_helper.cxx:69',
          'zsi_gtw_helper.cxx:2695',
          'zsi_gtw_handler.cxx:406',
```

```
            'zsi_gtw_handler.cxx:363',
            'ivr_igo_worker.cxx:968',
            'zsi_gtw_contact_helper.cxx:605',
            'mgmt_listener_thread.cxx:107',
            'zsi_gtw_contact_helper.cxx:605',
            'zsi_gtw_contact_helper.cxx:596',
            'zsi_gtw_handler.cxx:383',
            'gen_ils_main_handler.cxx:195',
            'zsi_gtw_contact_helper.cxx:234',
        ],
        IgnoreText => [
            qq|Can't find tan information in request|,
        ],
}
```

Name	Description
IgnoreSource	When component reports an error and the source of this error is on the **IgnoreSource** list, an alarm is not raised
IgnoreTest	When component reports an error and the error message contains string defined on the **IgnoreText** list, an alarm is not raised

Table 53. softax_ima configuration parameters

7.3.24. softax_ping

Overview

The probe monitors the Softax web based application ping. It raises an alarm when an application ping reports an error and when defined thresholds are exceeded.

Discover

Mixed

Test method

HTTP, SSL

Used parameters

Name	Type	Description
softax_ping_port	Mandatory	TCP port of the web server which servers the Softax application ping
softax_ping_protocol	Mandatory	Protocol used by a web server serving the Softax application ping; available values: **http**, **ssl**
threshold_high	Optional	Test duration reported by the Softax application ping high threshold (seconds); the default value is defined in the probe configuration file, **ThresholdHighDefault**
threshold_medium	Optional	Test duration reported by the Softax application ping medium threshold (seconds); the default value is defined in the probe configuration file, **ThresholdMediumDefault**
timeout	Optional	Application ping request timeout (seconds); the default value is defined in the probe

		configuration file, **DefaultTimeout**

Table 54 softax_ping parameters

Configuration

The probe configuration is kept in the **/akkada/etc/conf.d/Probes/softax_ping.conf** file.

```
{
      'DefaultTimeout' => 30,
      'DiscoverTestName' => 'ias_html',
      'ThresholdHighDefault' => 10,
      'ThresholdMediumDefault' => 5
}
```

Name	Description
DefaultTimeout	Application ping request default timeout (seconds)
DiscoverTestName	The name of the specific Softax application ping function used for detecting the Softax application ping
ThresholdHighDefault	Default test duration reported by the Softax application ping high threshold (seconds)
ThresholdMediumDefault	Default test duration reported by the Softax application ping medium threshold (seconds)

Table 55. softax_ping configuration parameters

7.3.25. ssl_generic

Overview

The probe monitors SSL sockets. If a script is defined, it executes the defined script on the socket. It raises an alarm when it cannot establish an SSL session, defined thresholds are exceeded or the script reports errors.

Discover

Automatic

Available SSL sockets are discovered by nmap. They can also be added manually.

Test method

SSL

Used parameters

Name	Type	Description
port	Mandatory	TCP port; set automatically; **do not change it manually**
ssl_generic_name	Mandatory	Service name reported by nmap during discovering process; set automatically; **do not change it manually**
ssl_generic_script	Optional	Scripts definition; see the section **Script syntax** in this subchapter
threshold_high	Optional	SSL socket opening duration high threshold

		(seconds), the default value configured in the probe configuration file, **ThresholdHighDefault**
threshold_medium	Optional	SSL socket opening duration medium threshold (seconds), the default value configured in the probe configuration file, **ThresholdMediumDefault**
Timeout	Optional	SSL socket opening timeout (seconds); default 3

Table 56 ssl_generic parameters

Set parameters

port, **ssl_generic_name**

Configuration

The probe configuration is kept in the **/akkada/etc/conf.d/Probes/ssl_generic.conf** file.

```
{
      'Table' => [
        443,
        465,
        563,
        636,
        993,
        995
      ],
      'nmap' => '/usr/bin/nmap --host_timeout 30000 -sS -p 1-513,515-543,545-1024',
      'ThresholdHighDefault' => 10,
      'ThresholdMediumDefault' => 5,
      'IgnoreMode' => 1,
      'DiscoverPreventFirewallFakes' => 12,
}
```

Name	Description
DiscoverPreventFirewallFakes	Some of the firewalls and IDS systems report all scanned ports are opened when they detect **nmap** scans; this is the maximum number of open ports reported by **nmap** which will be accepted during the discovery process; when the number of discovered ports exceeds this limit the whole result is ignored
IgnoreMode	1 = during discovering ignore detected open TCP ports if they **are NOT** on the **Table** list; 0 = during discovering ignore detected open TCP ports if they **are** on the **Table** list
nmap	Full path to the **nmap** binary with options; it has to end with option "-p" because during the discovery process, at the end of the configured value the list of ports from the **Table** list is added
Table	List of ports which need special treatment
ThresholdHighDefault	Default test duration high threshold (seconds)
ThresholdMediumDefault	Default test duration medium threshold (seconds)

Table 57. ssl_generic configuration parameters

Script's syntax

An asynchronous script can be defined with the **ssl_generic_script** parameter. If **ssl_generic_script** is defined, probe plays this script on the opened SSL socket and raises an alarm if it fails. The syntax of the script:

```
[key1]::[string2]||.. ..||[keyX]::[stringX]
```

where:

- key – **wait** or **send** (meaning: wait for a string or send a string)
- string – a string which is expected on the socked; a new line character must be defined in the following way: %NL%; expected string can be a regular expression.

There is no limit for the number of key-string pairs. When the script is defined, the service icon changes from to .

Example 1

This script obtains a web page /SJM/PL/WYN/W/index.htm and checks if there is a string "hello world" or "index.html" in the response:

```
send::GET /SJM/PL/WYN/W/index.htm HTTP/1.0%NL%%NL%| |wait::hello world|index.htm
```

Example 2

This script checks if the SMTP server answers correctly:

```
wait::220||send::HELO akkada.test.com||wait::250
```

7.3.26. tcp_generic

Overview

The probe monitors TCP sockets. If a script is defined, it executes the defined script on the socket. It raises an alarm when it cannot establish the TCP session, defined thresholds are exceeded or the script reports errors.

Discover

Automatic

Available TCP sockets are discovered by nmap. They can also be added manually.

Test method

TCP

Used parameters

Name	Type	Description
port	Mandatory	TCP port; set automatically; **do not change it manually**

tcp_generic_name	Mandatory	Service name reported by nmap during discovering process; set automatically; **do not change it manually**
tcp_generic_script	Optional	Scripts definition; see the **Script syntax** section, subchapter **ssl_generic** in this chapter – syntax is the same as in the case of the **tcp_generic** probe
threshold_high	Optional	TCP socket opening duration high threshold (seconds), the default value configured in the probe configuration file, **ThresholdHighDefault**
threshold_medium	Optional	TCP socket opening duration medium threshold (seconds), the default value configured in the probe configuration file, **ThresholdMediumDefault**
Timeout	Optional	TCP socket opening timeout (seconds); default 3

Table 58 tcp_generic parameters

Set parameters

port, tcp_generic_name

Configuration

The probe configuration is kept in the **/akkada/etc/conf.d/Probes/ssl_generic.conf** file.

```
{
      'Table' => [
        443,
        465,
        563,
        636,
        993,
        995
      ],
      'nmap' => '/usr/bin/nmap --host_timeout 30000 -sS -p 1-513,515-543,545-1024',
      'ThresholdHighDefault' => 10,
      'ThresholdMediumDefault' => 5,
      'IgnoreMode' => 0,
      'DiscoverPreventFirewallFakes' => 12,
}
```

Name	Description
DiscoverPreventFirewallFakes	Some of the firewalls and IDS systems report all scanned ports are open when they detect **nmap** scans; this is the maximum number of open ports reported by **nmap** which will be accepted during the discovery process; when the number of discovered ports exceeds this limit whole result is ignored
IgnoreMode	1 = during discovering ignore detected open TCP ports if they **are NOT** on the **Table** list; 0 = during discovering ignore detected open TCP ports if they **are** on the **Table** list

nmap	Full path to the **nmap** binary with options; it has to end with option "-p" because during the discovery process, at the end of the configured value the list of ports from the **Table** list is added
Table	List of ports which need special treatment
ThresholdHighDefault	Default test duration high threshold (seconds)
ThresholdMediumDefault	Default test duration medium threshold (seconds)

Table 59. ssl_generic configuration parameters

7.3.27. tcpip

Overview

The probe monitors the TCP/IP stack. The main purpose of this probe is to collect statistics but it raises an alarm when thresholds are defined and exceeded (by default it doesn't raise any alarms).

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
tcpip_icmpInErrors_threshold_percent	Optional	Number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.; percentage threshold (%)
tcpip_icmpInErrors_threshold_units	Optional	Number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.; units threshold (number of packets)
tcpip_icmpOutErrors_threshold_percent	Optional	Number of ICMP messages which this entity did not send due to problems discovered within ICMP, such as lack of buffers. This value should not include errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations, there may be no types of error which contribute to this counter value; percentage threshold (%)
tcpip_icmpOutErrors_threshold_units	Optional	Number of ICMP messages which this entity did not send due to problems discovered within ICMP, such as lack of buffers. This value should not include

		errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations, there may be no types of error which contribute to this counter value; unit threshold (number of packets)
tcpip_ipFragFails_threshold_units	Optional	Number of IPv4 datagrams that have been discarded because they needed to but could not be fragmented at this entity, e.g., because their Don't Fragment flag was set; unit threshold (number of packets)
tcpip_ipInAddrErrors_threshold_percent	Optional	Number of input datagrams discarded because the IPv4 address in their IPv4 header destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IPv4 routers, and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address; percentage threshold (%)
tcpip_ipInAddrErrors_threshold_units	Optional	Number of input datagrams discarded because the IPv4 address in their IPv4 header destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IPv4 routers, and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address; unit threshold (number of packets)
tcpip_ipInHdrErrors_threshold_percent	Optional	Number of input datagrams discarded due to errors in their IPv4 headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IPv4 options, etc.; percentage threshold (%)
tcpip_ipInHdrErrors_threshold_units	Optional	Number of input datagrams discarded due to errors in their IPv4 headers, including bad checksums, version number

		mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IPv4 options, etc.; unit threshold (number of packets)
tcpip_ipReasmFails_threshold_units	Optional	Number of failures detected by the IPv4 re-assembly algorithm (for any reason: timed out, errors, etc). Note that this is not necessarily a count of discarded IPv4 fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received; unit threshold (number of packets)
tcpip_tcpInErrs_threshold_percent	Optional	Total number of segments received in error (e.g., bad TCP checksums); percentage threshold (%)
tcpip_tcpInErrs_threshold_units	Optional	The total number of segments received in error (e.g., bad TCP checksums); unit threshold (number of packets)
tcpip_threshold_percent	Optional	Default percentage threshold for any kind of monitored errors (%); specific percentage threshold overwrites this value
tcpip_threshold_units	Optional	Default units threshold for any kind of monitored errors (%); specific unit threshold overwrites this value
tcpip_udpInErrors_threshold_percent	Optional	Number of received UDP datagrams that could not be delivered for reasons other than lack of an application at the destination port; percentage threshold (%)
tcpip_udpInErrors_threshold_units	Optional	Number of received UDP datagrams that could not be delivered for reasons other than lack of an application at the destination port; unit threshold (number of packets)

Table 60 tcpip parameters

7.3.28. ucd_ext

Overview

The probe monitors UCD (Net-SNMP, standard SNMP agent at Linux platforms) script extensions. The probe raises an alarm when UCD extension reports an error, an error is reported by extension script in the AKK@DA format, script output doesn't match an expected value, the UCD extension is not available or script output value exceeds defined thresholds.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
ucd_ext_bad	Optional	Unexpected string in a script output (string); if exists in the script output, an alarm is raised; used only when the **raw** format is detected;
ucd_ext_data_type	Mandatory	Detected format type; set automatically; **do not change it manually**
ucd_ext_expect	Optional	Expected string in a script output (string); if it doesn't exist in the script output, an alarm is raised; used only when the **raw** format is detected;
ucd_ext_max	Optional	Maximum threshold for the value reported by an extension script (number); used only when the **raw** format is detected;
ucd_ext_min	Optional	Minimum threshold for the value reported by an extension script (number); used only when the **raw** format is detected;

Table 61 ucd_ext parameters

Set parameters

ucd_ext_data_type

Net-SNMP agent extensions configuration

E.g. external script **mail_queue.sh** for monitoring the Sendmail mail queue can be attached to the Net-SNMP agent configuration file **/etc/snmpd.conf** by adding the following line:

```
exec   "mail_queue" /usr/local/bin/mail_queue.sh
```

mail_queue.sh returns the current number of mails in the Sendmail mail queue. Its content:

```
#!/usr/bin/perl
my $a = `/usr/bin/mailq | grep Total`;
$a =~ s/\s+//g;
$a = (split /\:/, $a)[1];
print "$a\n";
```

For more information regarding Net-SNMP agent extensions refer to the manual of the **snmpd.conf** file.

Formats' syntax

The probe supports three types of format which can be produced by UCD extension scripts – **raw**, the AKK@DA **text** format and the AKK@DA **stat** format. These formats are detected automatically. It is strongly recommended to use AKK@DA formats instead of the **raw** format. When the **raw** format is used it's not possible to inform a/the probe how to interpret data given by UCD extension scripts and all thresholds and expected/unexpected strings have to be configured manually in the

specific service configuration. In the case of AKK@DA formats, the UCD extension script output provides full information regarding interpreting the data, thresholds, etc.

Raw

Raw is the most basic format. Raw format means the UCD extension script returns a substantial numeric or text value which should be interpreted by the probe as a number or a text. An example of the script which returns this kind of output is described in section **Net-SNMP agent extensions configuration**, script **mail_queue.sh**.

When the **raw** format is used, parameters described in the **Uses parameters** section are interpreted and can be used to tune the way how the probe raises alarms in a specific case. E.g. if it's needed to raise an alarm when **mail_queue.sh** shows there are more than 100 e-mails in the Sendmail mail queue, parameter **ucd_ext_max** should be set to 100.

When the **raw** format is used and the UCD extension script returns numeric values the probe automatically collects statistics into RRD databases.

The AKK@DA text format

This format should be used to report text outputs by UCD extension scripts to AKK@DA. Format definition:

```
AKKADA||TEXT||field1=xxx::field2=yyy…
```

Available fields:

- **output** – mandatory; single; any string which has to be interpreted by the probe
- **expected** – optional; multiple; a string which is expected in the **output** string; if the **expected** string is missing in the **output** string, an alarm is raised; if multiple **expected** strings are defined all of them have to appear in the **output** string otherwise an alarm is raised
- **bad** – optional; multiple; a string which is unexpected in the **output** string; if the **bad** string exists as a part of the **output** string, an alarm is raised; if multiple **bad** strings are defined, any **bad** string found as a part of the **output** string raises an alarm
- **brief** – optional; a string displayed in the **brief** column while the **detailed** view is used in the web GUI

E.g. the **arp_watch.sh** script checks if the given IP address is available on the network from the ARP protocol perspective:

```
#!/bin/sh
```

```
if [ "$1" = "" ] ; then
    echo "usage: arp_watch.sh <ip address>"
    exit 1
fi

ARP=`/sbin/arp -d $1 2>/dev/null; /bin/ping -c 1 -W 1 $1 >/dev/null; arp -a |
/bin/grep $1 | /bin/awk '{print $4}'`

echo "AKKADA||TEXT||bad=<incomplete>::output=$ARP::brief=ip address $1, MAC address
$ARP::errmsg=ip address unreachable"
```

The arp_watch.sh script uses the AKK@DA text format and raises an alarm when the MAC address for the given IP address is not found.

The AKK@DA stat format

This format should be used to report numeric outputs by UCD extension scripts to AKK@DA. Format definition:

```
AKKADA||STAT||field1=xxx::field2=yyy…||field1=xxx::field2=yyy…
```

Multiple numeric values can be reported in separated || sections. In each section the following fields are available:

- **output** – mandatory; single; any number
- **title** – mandatory; single; a track name in the sense of the RRD database (for more details regarding RRD, see the RRD manual); must be 1 to 19-character long using the following characters [a-zA-Z0-9_]
- **cfs** – mandatory; single; an RRD track type in the sense of the RRD database (for more details regarding RRD, see the RRD manual); e.g. COUNTER, GAUGE, ABSOLUTE
- **min** – optional; single; minimum threshold for the value given by the **output** field
- **max** – optional; single; maximum threshold for the value given by the **output** field

E.g. the **postfix_mail_queue_ad.sh** script collects the Postfix mail queue statistics regarding active and deferred e-mails in the mail queue.

```
#!/bin/sh

CMD=`which postconf 2>/dev/null`

if [ "$CMD" = "" ]; then
        echo "postconf command not found"
        exit 1;
fi

cd /etc/postfix
```

```
sum=0
qdir=`$CMD -h queue_directory`
active=`find $qdir/incoming $qdir/active $qdir/maildrop -type f -print | wc -l | awk
'{print $1}'`
deferred=`find $qdir/deferred -type f -print | wc -l | awk '{print $1}'`
echo
"AKKADA||STAT||title=Active::output=$active::cfs=GAUGE||title=Deferred::output=$defe
rred::cfs=GAUGE"
```

7.3.29. ucd_process

Overview

The probe monitors processes reported by the Net-SNMP agent (the standard SNMP agent at Linux platforms). The probe raises an alarm when: Net-SNMP reports an error, process information is not available or defined thresholds are exceeded.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
ucd_process_max	Optional	Maximum expected process count; if not defined or set to 0 this condition is not checked
ucd_process_min	Optional	Minimum expected process count; if not defined or set to 0 this condition is not checked

Table 62 ucd_process parameters

7.3.30. windows_service

Overview

The probe monitors Microsoft Windows services. It raises an alarm when service information is not avaiilable or service status is other than **active**.

Discover

Automatic

Test method

SNMP

Used parameters

Name	Type	Description
index	Mandatory	Service index; set automatically; **do not change it manually**
windows_service_hex	Optional	set automatically if a service name is reported by the SNMP agent using the HEX string format; **do not change it manually**

Table 63 ucd_process parameters

Set parameters

index, windows_service_hex

Configuration

The probe configuration is kept in the /akkada/etc/conf.d/Probes/windows_service.conf file. In this file there is configured the list of service names called **DiscoveryExclude** which has to be ignored during the discovery process.

When services are discovered and you realized there are other services (except for the ones listed on the **DiscoveryExclude** list) which you don't want to be discovered, just add their names to that list and restart the **nm-discovery.pl** process. If you realize you don't want to monitor some of the already discovered services, you can also add their names to the **DiscoveryExclude** list, restart the **nm-discovery.pl** process and then use the **/akkada/bin/windows_service_dus.pl** command line tool to quickly remove unwanted services from the AKK@DA configuration.

Chapter 8

8. Using the web GUI of AKK@DA

8.1. Displaying entities

There are 3 types of entities in AKK@DA: groups, nodes and services. The role of a group is to group nodes. A node represents a specific monitored host and has services. A service represents a specific service or resource of a monitored host (CPU, HDD, specific process, etc.). Information about groups, nodes and services is available via the AKK@DA web GUI. The access path is the same in all three cases. The only difference is the information which is presented in each of these 3 cases.

Overview

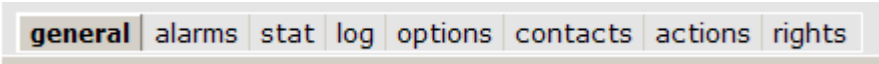
A group entity shows summary information about all contained hosts. That includes name, current status, vendor icon, functions on the network icon (e.g. router, switch, firewall, etc.), the age of collected data and some general ICMP stats.

A node entity shows general information about a specific host as described in the case of a group entity. Additionally, it shows some more detailed information about the host, such as IP forwarding, location, uptime and description. It also shows a list of all host services with summary information about each of them. This always includes a type of service, its status, name and the age of data. The rest of the information depends on the type of the specific service.

A service entity shows detailed information about a specific service monitored by AKK@DA.. It always depends on the type of the specific service.

Sections of information

Information regarding each entity is grouped in a few different groups, which are available via tabs.



General – contains all the collected text/numbers information.

Alarms – shows all alarms related to the specific entity and all its children entity alarms (in the sense of **location tree**). This means in the case of a service only its alarm is shown. In the case of a node there are shown alarms of all node services. In the case of a group there are shown alarms of all services of all nodes which belong to the group .

Stat – exists only for the case of nodes and services. It shows collected data on charts. In the case of a node entity it shows node graphs and default graphs of all of its services. In the case of a service entity it shows all service graphs.

Options – shows entity options and allows the configuring of a specific entity.

Contacts – shows contact groups connected to an entity and allows connecting them to/ disconnecting them from entities

Actions – shows actions (like sending alerts, executing scripts) connected to an entity and allows the connecting/disconnecting of them.

Rights – shows current user rights to a specific entity.

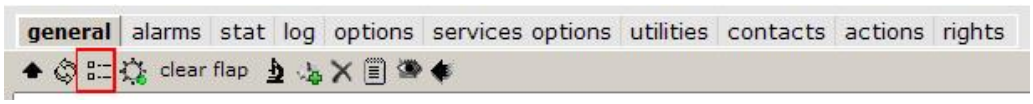
Accessing information

There are a few ways of accessing entities information. By clicking a specific object from the **location tree** you can access a group or node entity. Information is always presented in the right window. To see service information first click its node on the **location tree** and then click the name of the required service in the right window.

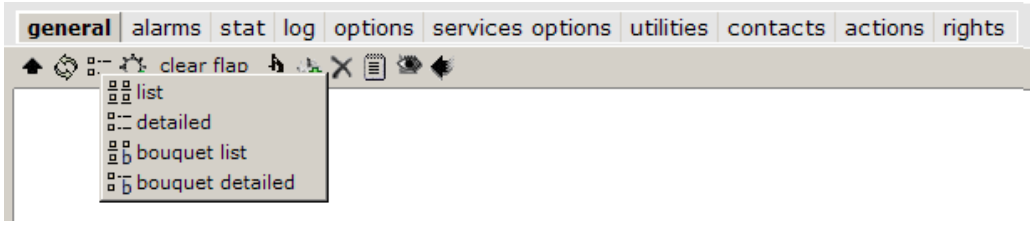
Accessing information as described points you to a current section. e.g. if you are in the **general** tab in the right window, you will be pointed to the **general** tab of the chosen entity. If you prefer to be pointed to the other section at once, right click the specific entity and from the context menu select the section you need.

Nodes' views

While you are accessing a node entity there are 4 options of displaying information. They are available under the **view** icon (in red below):

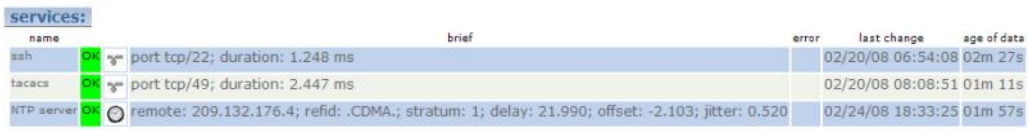


Click this icon for choosing the required **view**:



The **list** view shows much summarized information about all node services which includes only a name, status and description if available. It is useful for quick estimation of host problem

The **detailed** view shows detailed summary information about all node services in a table manner. This is the most detailed summary view available in AKK@DA. It allows fast checking of the current values/levels/durations/etc. of node services.



Views **bouquet list** and **bouquet detailed** work as described above **list** and **detailed** except from the fact they show group information by type of services. They are useful when a node has a lot of services. They show a number of services of a specific type having specific status.

services:

probe name	status	entities statuses
CPU	OK	OK (1)
fan	OK	OK (3)
network interface	Down	OK (26) Down (1) No status (24)
NTP	OK	OK (1)
power supply	OK	OK (2)
RAM	OK	OK (1)
TCP socket	OK	OK (1)
TCP/IP statistics	OK	OK (1)
temperature sensor	OK	OK (1)

Tree

The **location tree** in the left panel by default displays groups and nodes configured in AKK@DA. This tree is based on JavaScript so the whole displayed structure is built on the client's web browser side. This may cause performance issues if the client's computer has too little resources or when AKK@DA monitors a lot of hosts. In such cases the displaying of node entities should be disabled to speed up browsing the AKK@DA GUI. This can be done by modifying the **/akkada/etc/conf.d/Web/Tree.conf** file, the **GroupMode** key should be set to 1. After changing that, the Apache server must be restarted to implement the change. In this limited mode only group entities and node entities which belong to the currently browsed group are displayed.

8.2. Searching entities

Searching

To search entities in the left panel click tab **find**, fill in the searching conditions and then click button **find**. There are a couple of searching criteria. Not all of them are mandatory, however, at least one condition must be filled in. They can be mixed as needed, there are no restrictions.

entity name – full or partial entity name,

entity IP – full or partial entity IP,

case sensitive – by default all conditions are not case sensitive; enabling this option enables case sensitive policy for all conditions,

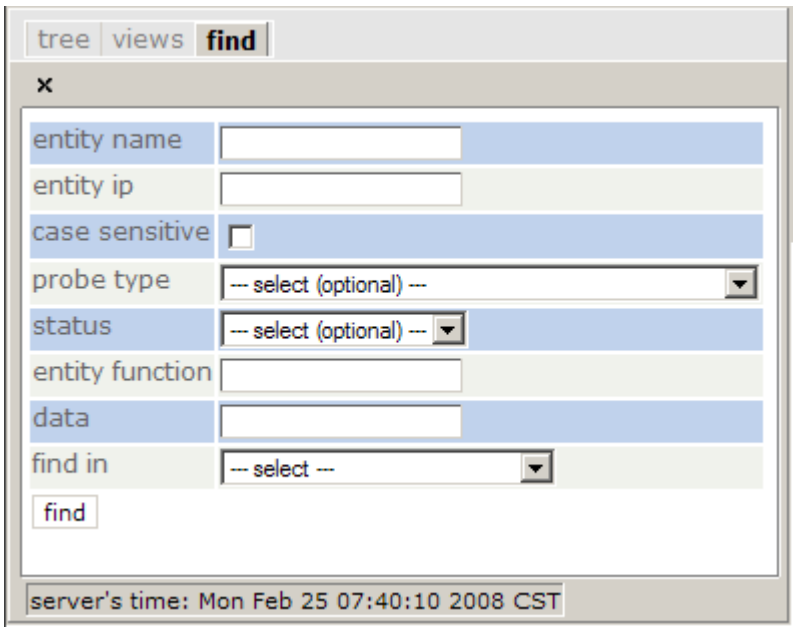
probe type – restricts searching to entities of a chosen type,

status – restricts searching to entities which have chosen status,

entity function – restricts searching to entities of a typed in function (e.g. router, switch, etc.),

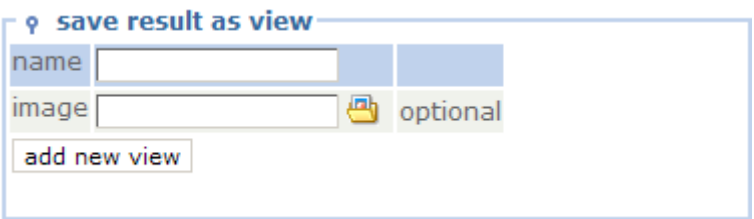
data – this is the most powerful and the most difficult field to use; most of the information collected by AKK@DA is kept in files (each entity has its own file); these files are stored in a directory configured in the **/akkada/etc/conf.d/Probe.conf**, file key **DataDir**; information stored in this file is in text format, but it's raw; its meaning may be difficult to understand; this option is used to search entities by information stored in these files; it's recommended to see the content of some of these files for a better understanding before starting to use this field; simple regular expressions can be used,

find in – restricts searching to a chosen part of the **location tree**.



Saving results

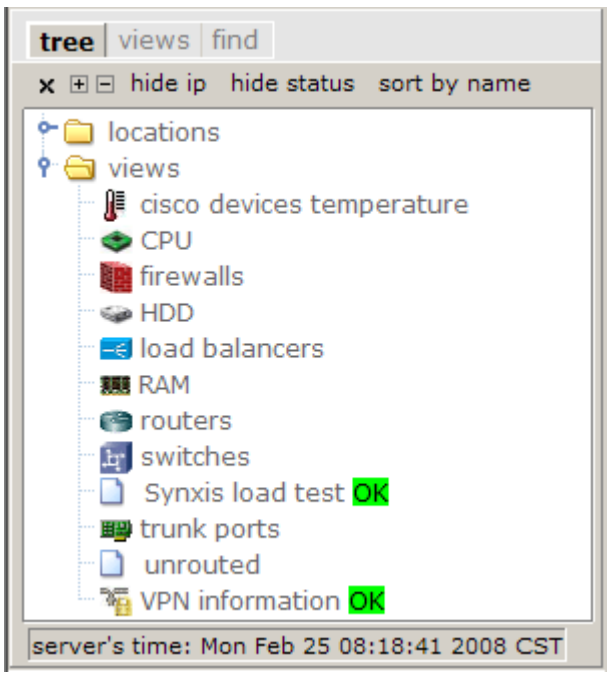
To save the search result as a dynamic view (described in the **Using views** chapter) use the form available in the left panel when the reach result is available. The **name** field is mandatory, **image** is optional.



8.3. Using views

Overview

Views are an alternative way of organizing entities in the AKK@DA web GUI. As known from the previous chapter there are groups which have nodes and these nodes contain services. This structure is called the **location tree**. Views allow organizing entities independently of the **location tree**. A view is a container which can group any entities in one place. It is possible to have all services of the same type as a single view – e.g. all CPUs. It is possible to have group, node and service entities in a single view. There are no restrictions. A view shows information about its members. e.g. if there is a view with all monitored CPUs, inside of this view it's possible to see summary details of all CPUs, all their alarms, log messages and default charts. Views are accessible via the left panel, in the **views** folder.

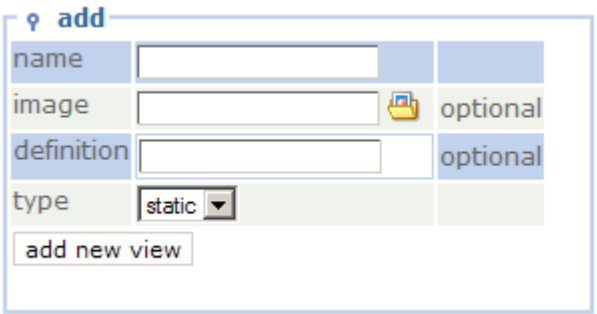


Types of views

There are two types of views – **static** and **dynamic**. **Static** views have to be created manually by the user. It can be done by adding each needed entity to the view. **Dynamic** views are like saved search results – they contain entities which meet selected criteria. **Static** views are hard to manage, but they have property **status** which shows their calculated status. This status is a result of the status of all its members. **Dynamic** views don't have property **status**, but they can be quickly created and they are easy to manage.

Static views

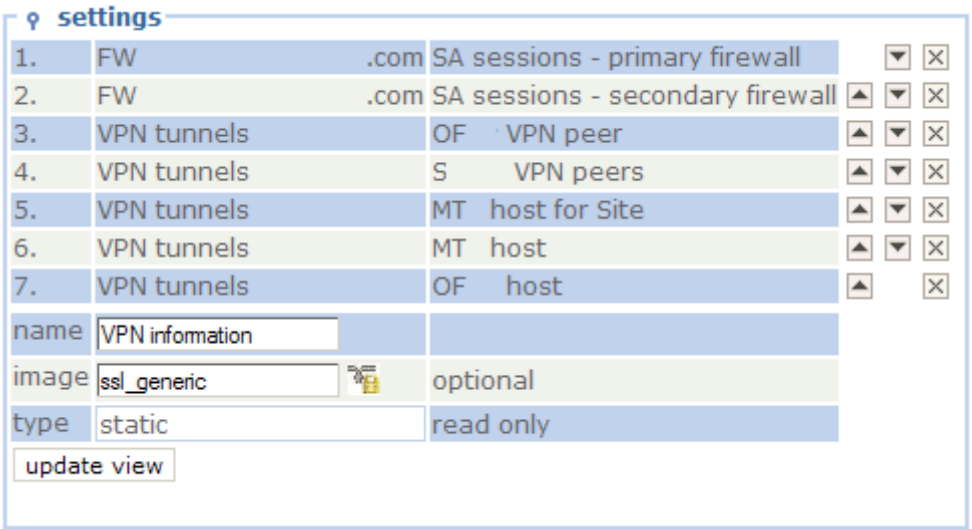
To create a static view, click tab **views** in the left panel, fill in the **add** form and click button **add new view**.



The **name** field is mandatory. The **type** must be **static**. In this case the **definition** field is ignored.

After it has been done, for each entity needed in this view, right click and select the **add to view** menu from the context menu. From the form which will appear at the top of the right panel select the needed view and click button **add**.

For managing static views use the left panel, tab **views**, form **settings**.

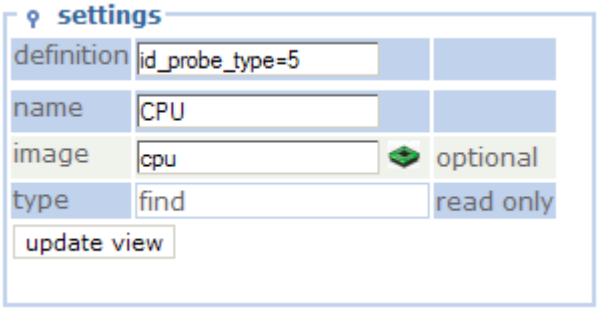


The name and the image of the view can be modified by using this form. Also the order of entities inside the view can be changed and selected entities can be removed from the view.

Dynamic views

The easiest way to **create a dynamic view** is saving a search result as a dynamic view. This process is described in the **Searching entities** chapter, section **Saving results**. The dynamic view can also be created manually by using the **add** form in the left panel, tab **views** (see section **Static views** above). In this case the **type** field must be set to **find** and field **definition** is mandatory. To understand what needs to be typed into the

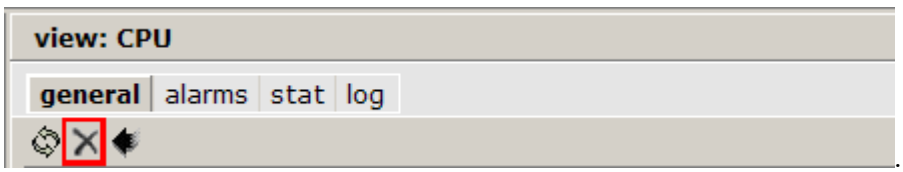
definition filed save any searching reasult as a dynamic view and then browse its properties. For **managing a dynamic view** use the left panel, tab **views**, form **settings**.



All fields but the **type** field can be modified.

Deleting views

To delete a view, click the view in the left panel. When the view is loaded, in the right panel click the **delete** icon and confirm the operation.



8.4. Managing contacts

Overview

Contacts are used for two reasons. Firstly, they are to know who is responsible for a monitored entity, which is important information if something goes wrong with this entity. In other words, they are to allow AKK@DA users to inform owners of monitored entities that they have a problem. This chapter describes how to use contacts for this purpose and how to manage contacts. Secondly, they are utilized by the actions subsystem for addressing notifications about detected faults of the monitored entities to the defined people. This is described in the **Managing actions** chapter.

Almost all configuration options regarding contacts are available through the AKK@DA web GUI, top menu, button **contacts**.

Using

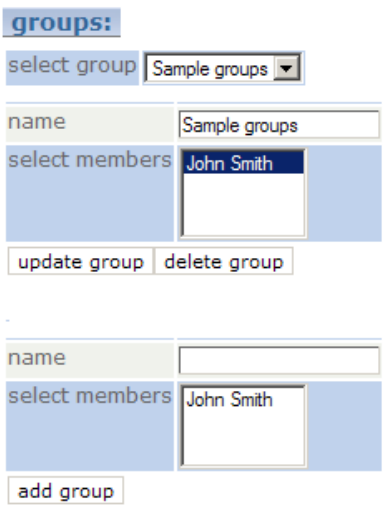
When any contact information is bound to the entity, AKK@DA shows a phone icon in the web GUI. This icon is almost always presented while this entity is being

browsed. Additionally it is also presented with an alarm if any alarm regarding this entity is raised. Clicking this icon causes applicable information to be shown.

Character ***** at the beginning means that contact is directly connected to this entity. If there is no star at the beginning, contact is inherited from some other entity by the **location tree** structure.

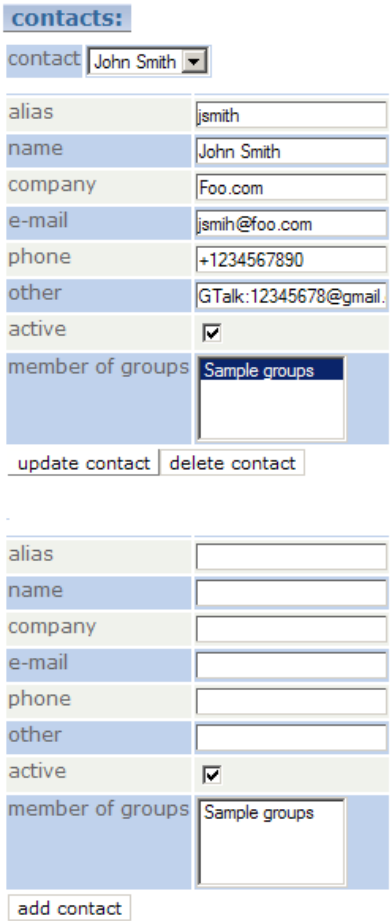
Contact groups

Contact groups are containers which contain specific contacts. AKK@DA uses only contact groups. There is no option to use a specific contact directly. **To create** a contact group, use the bottom form of section **groups**. Group **name** is a mandatory field. Contact group members can also be chosen optionally. **For updating** contact group members or **deleting** a contact group, use the top form of the section **groups**.



Contacts

Contacts are records which describe specific people. Each contact should be a member of at least one **contact group** otherwise it will be useless. **To create** a contact, use the bottom form of the section **contacts**.



Fields **alias** and **name** are mandatory, others are optional. **Alias** must be unique.

In the **group members** field at least one group should be selected, otherwise the contact won't be used.

The **active** field enables/disables a contact. When the contact is disabled it is not used for any purposes by AKK@DA. This option is for temporarily disabling contacts (e.g. if someone is on vacation, you can disable their contact record to protect them from being called in the case of any problem regarding the systems they manage).

Fields **company** and **phone** are only for providing information.

Fields **e-mail** and **other**, except for containing information, are used by the action subsystem for sending notifications. The **e-mail** field is just a person's e-mail. The **other field** should contain information used by the specific action subsystem (see chapter **Managing actions** for more details). If actions are not used, **other** should stay empty.

For updating or **deleting** contact, use the top form of the section **contacts**.

Binding contacts

When contacts and contact groups are created, they have to be connected to applicable entities, otherwise they are useless. This process is called **binding**.

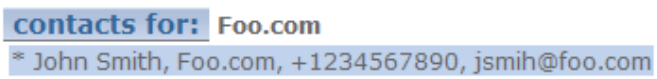
Contacts in AKK@DA are inherited based on the **location tree** structure. If you bind a **contact group** to the entity, this **contact group** information is inherited to all entities in the selected sub tree. There are no restrictions for bindings. Contact groups can be bound to any entity, at any level of the **location tree**. Any number of contact groups can be bound to one entity. e.g. if you monitor a few web servers managed by the same team, put these servers into one group in the **location tree** and bind the applicable contact group to this group – contact group information will be inherited to all of these servers.

There are two ways of binding.

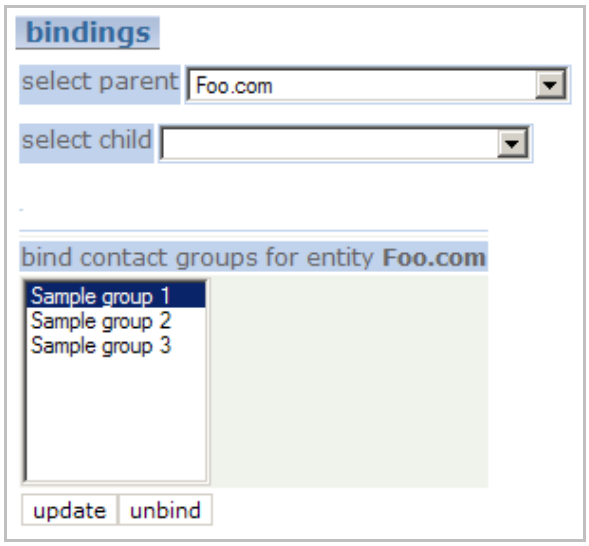
To bind a contact group to an entity, open the entity, section **contacts** (see chapter **Displaying entities**).

From the form select the contact groups you need to bind and click the **update** button. To remove all bindings click the **unbind** button.

When there are any contact groups bound to the selected entity, a preview of the current contact information which will be shown by the web GUI is presented in the same place.



Bindings can also be managed through **top menu**, button **contacts**, section **bindings**. The idea of binding is still the same, but in this way it is possible to create bindings for many entities in one place, which is sometimes faster.



Select parent is mandatory. This is for selecting an entity for binding.

Select child is optional. If it's needed to bind a contact group to a specific service entity, this service can be selected by this field. When select **child** is not empty, selected contact groups are bound to the entity chosen in the **select child** field, not to the entity indicated by the **select parent** field.

Integration with external systems

AKK@DA allows integration of its contact group model with external web based services. It is possible to add extra information provided by an external system to any contact.

E.g. if you have a few people working on the on-call rota and their shifts are managed by any information system, this system can be integrated with AKK@DA to display contact availability information for each specific contact.

Integration with an external system requires a special module to be developed by a user. An example module **OnCall.pm** is provided by AKK@DA release. The OnCall.pm module requires the external system to provide:

- data through a web or a web service channel,
- data to be in the XML standard.

For the external system to be used it needs to be configured in the **/akkada/etc/conf.d/Web/Contacts.conf** file:

```
{
  ExternalInformation => {
    'Enabled' => 1,
    'URL' => 'http://10.13.77.250/oncall/available/%s/today/xml',
    'Module' => 'OnCall',
  },
},
```

where **%s** in the **URL** definition will be exchanged with contact alias from the AKK@DA contact database. Parameter **Module** is the name of the Perl module which will process information fetched by AKK@DA from the source defined in the **URL** parameter. This module should contain function **process** which should return a string result. This result will be attached to the related AKK@DA contact record as first information.

An example **OnCall.pm** module:

```
package OnCall;

use vars qw($VERSION);

$VERSION = 0.1;

use strict;

sub process
{
    my $ref = $_[1];
    my $result = '';
    my $class = 'm';

    return [$result, $class]
        unless defined $ref->{oncall};

    if ($ref->{oncall}->{available} eq "true")
    {
        $result = '<img class="b10" src="/img/on.gif" alt="on call">';
    }
    elsif ($ref->{oncall}->{available} eq "false")
    {
        $result = '<img class="b10" src="/img/off.gif" alt="not available">';
    }

    return [$result, $class];
}

1;
```

8.5. Managing actions

Overview

The action subsystem is a group of mechanisms used by AKK@DA for sending notifications and executing external applications in the case of detecting any issue related to the monitored environment. e.g. it can be used to inform administrators by e-mail or instant message when the server managed by them is unreachable. Currently the action subsystem supports sending notifications by e-mail, GTalk and Gadu-Gadu (a Polish instant messaging service). Before you start configuring actions make sure you have configured the contact groups (see chapter **Managing contacts**).

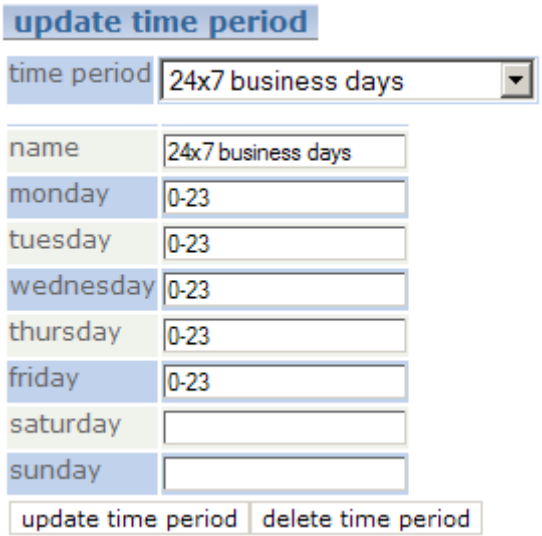
The architecture of the action subsystem contains the following objects:

- **contact group** – defines addresses of people who should be informed,
- **time period** – defines when users wish to have actions running,
- **command** – defines what exactly should be done (e-mail, IM, etc.),
- **action** – uses the **command** object; defines a specific action; e.g. it says: when a node is unreachable, after 5 minutes it sends an e-mail notification and repeats it every 30 minutes until the node is back up;
- **binding** – puts together **action**, **time period** and **contact group** objects and binds them into a specific entity; in other words, it enables a specific action on a specific entity.

Time periods

The action subsystem detects whether or not something should be done based on a few criterias. One of them is a time policy defined by the **time period** object. An action configured in AKK@DA must have relation with the time period object. The **Time period** object defines the days of the week and hours when the action **can** be executed. Day/time decision is always made by using local AKK@DA's server time. There are no options to configure time zone in the **time period** object.

To manage time periods, in **top menu**, click button **actions** and then click the **time periods** tab. To add a new time period, use the **new time period** form. To update or delete an existing time period, use the **update time period** form. The meaning of the fields in both forms is exactly the same.



The name field is mandatory and must be unique. It's only a name and can be anything.

Monday-Sunday fields are optional and should contain hours, when the specific action is expected to be running An empty value means no actions for a given day. Otherwise hours should be defined in a 24h manner. Hours can be comma separated. Ranges of hours can be used with "-" characters. 0 means 00:00-00:59, 23 means 23:00-23:59.

For example:

0-23 – means all day, 24h,

2,6,18 – means 2:00-2:59 and 6:00-6:59 and 18:00-18:59,

0-9,17-23 – means 0:00-9:59 and 17:00-23:59.

Commands

Command objects are definitions which inform the action subsystem what exactly should be done. Currently there are 3 command modules available in AKK@DA: **mail**, **GTalk** and **gg**. But there is no limit on configured command objects. There can be multiple command objects using e.g. module **mail**, but with different parameters.

To manage command, in **top menu**, click the **actions** button and then click the **commands** tab. To add a new command, use the **new command** form. To update or delete an existing command, use the **update command** form. The meaning of the fields in both forms is exactly the same.

Name field is mandatory and must be unique. It's only a name and can be anything.

Module field is mandatory. A specific command module must be chosen.

Command field is mandatory too, but its content depends on the selected module and is described later in this chapter.

E-mails

E-mail notification uses the **mail** module whose configuration is stored in the **/akkada/etc/cond.f/ActionsExecutor/mail.conf**: file

```
{
    SMTP => '127.0.0.1',
    'Defaults' => {
        'from' => 'akkada',
        'subject' => '%%PARENT_NAME%%, %%ENTITY%%: %%STATUS%%',
    },
}
```

Name	Description
SMTP	IP address of the SMTP gateway which AKK@DA uses to send e-mails. To avoid blocking the action subsystem it is recommended to use the local SMTP server installed on AKK@DA's server for sending e-mail to recipients. Using an external SMTP server may cause the action subsystem will be blocked or there will be delays in executing any type of actions if the external SMTP server is unavailable or responces slowly.
from	Default value of field **from** of the e-mail envelope
subject	Default value of field **subject** of the e-mail envelope; the following variables related to the affected entity can be used: **%%DESC%%** - entity description, **%%ENTITY%%** - entity name, **%%ERRMSG%%** - alarm description,

	%%PARENT_NAME%% - entity parent name, **%%STATUS%%** - entity status.

After changing anything in the file described above, signal **HUP** should be sent to the **nm-actions_executor.pl** process to reload configuration.

When it has been configured, it's needed to create a **command** object in AKK@DA which will send e-mails. The way to do it was described earlier in this chapter (except for the content of the **command** field). The content of the **command** field is strict. The syntax is as follows:

```
field1 => "value1", field2 => "value2"
```

Available field names (case sensitive) are: **smtp**, **from**, **subject**, **cc, bcc**. All of them are optional. If they are defined, they overwrite values defined in the **/akkada/etc/conf.d/ActionsExecutor/mail.conf** file.

E.g. use **from => "akkada\@localhost"** to overwrite the default value of the **from** field configured in the **mail.conf** file.

GTalk

AKK@DA can send notification using the Google GTalk instant messaging service. To use this feature it's needed to have a dedicated GTalk user account for AKK@DA (a normal GTalk user account which will be used by AKK@DA to send instant messages to recipients). For this purpose AKK@DA uses the **gtalk** module whose configuration is stored in the **/akkada/etc/cond.f/ActionsExecutor/gmail.conf**: file

```
{
    'Defaults' => {
        username => '',
        password => '',
    }
}
```

Name	Description
username	Default GTalk user name (GTalk user who will send messages)
password	Default GTalk user (defined by the **username** field) password

After changing anything in the file described above, signal **HUP** should be sent to the **nm-actions_executor.pl** process to reload configuration.

When this has been configured, it's needed to create a **command** object in AKK@DA which will send e-mails. The way to do it was described earlier in this chapter (except for the content of the **command** field). The content of the **command** field is strict. The syntax is as follows:

```
username => "value1", password => "value2"
```

If username and password are defined here, they overwrite values defined in the **/akkada/etc/conf.d/ActionsExecutor/GTalk.conf** file.

Using GTalk notification needs some configuration of the contact records. People who should be alarmed via GTalk have to have the **other** field configured in their contact records (see the **Managing contacts** chapter). **Other** fields should contain a string in the following format:

```
GTalk:<GTalk's ID>
```

E.g. GTalk:12345678@gmail.com

If a **contact** object has no string starting with **GTalk::** in field **other**, Gtalk notifications are not send to this conntact. The **other** field is case sensitive.

Gadu-Gadu

AKK@DA can send notification using the Gadu-Gadu instant messaging service (this is a Polish local IM service). To use this feature it's needed to have a dedicated Gadu - Gadu user number for AKK@DA (a normal Gadu-Gadu user number which will be used by AKK@DA to send instant messages to recipients). AKK@DA uses the **gg** module whose configuration is stored in the **/akkada/etc/cond.f/ActionsExecutor/gg.conf**: file

```
{
    'Defaults' => {
        username => '',
        password => '',
    }
}
```

Name	Description
username	Default Gadu-Gadu user number (a Gadu-Gadu user who will send messages)
password	Default Gadu-Gadu user (defined by the **username** field) password

After changing anything in the file described above, signal **HUP** should be sent to the **nm-actions_executor.pl** process to reload configuration.

When this has been configured it's needed to create a **command** object in AKK@DA which will send e-mails. The way to do this was described earlier in this chapter (except for the content of the **command** field). The content of the **command** field is strict. The syntax is as follows:

```
username => "value1", password => "value2"
```

If username and password are defined here, they overwrite values defined in the **/akkada/etc/conf.d/ActionsExecutor/gg.conf** file.

Using Gadu-Gadu notification needs some configuration of contact records. People who should be alarmed via Gadu-Gadu have to have the **other** field configured in their contact records (see the **Managing contacts**). chapter The **Other** field should contain a string in the following format:

```
gg:<Gadu-Gadu ID>
```

E.g. gg:12345678

If a **contact** object has no string started with **gg::** in field **other**, Gadu-Gadu notifications are not send to this conntact. The **other** field is case sensitive.

Actions

Action objects define what has to be done (see the **commands** section above) and what conditions have to be met to do it.

To manage actions, in top menu, click the **actions** button and then click the **actions** tab. To add a new command, use the **new action** form. To update or delete an existing command, use the **update action** form. The meaning of the fields in both forms is exactly the same.

new action:

name	
notifications interval	1800
notifications start after	120
notifications count	4
notify recovery	☑
service type	
error messages like	
statuses	1-6
ignore calculated status changes	☑
inherit to children	☑
command	--- select ---
active	☑

add action

The **name** field is mandatory and must be unique. It's only a name and can be anything.

The **notification interval** field is mandatory. It must be a number greater than 0. It defines intervals in seconds between following notifications.

The **notification start after** field is mandatory. It must be a number. It defines a delay in seconds before the first notification will be sent. If it's set to 0, the first notification will be sent immediately.

The **notification count** field is mandatory. It must be a number greater than 0. It defines the number of notifications which will be sent. Delay between subsequent notifications is defined by the **notifications interval** field.

Notify recovery defines if the recovery notification will be sent when the entity returns to OK status.

Service type – currently not used; for future use.

Error messages like – currently not used; for future use.

The **status** field is mandatory. It defines which status of entity has to send a notification. Syntax: numbers must be separated by commas and ranges by "-". e.g. 1,2,3 or 1-3 or 1,2-3. Numbers are defined in files **`/akkada/lib/Constants.pm`**:

```
#STATUSES
use constant _ST_OK          => 0;
use constant _ST_WARNING     => 1;
use constant _ST_MINOR       => 2;
```

```
use constant _ST_MAJOR       => 3;
use constant _ST_DOWN        => 4;
use constant _ST_NOSNMP      => 5;
use constant _ST_UNREACHABLE => 6;
use constant _ST_UNKNOWN     => 64;
use constant _ST_RECOVERED   => 123;
use constant _ST_INIT         => 124;
use constant _ST_INFO        => 125;
use constant _ST_BAD_CONF    => 126;
use constant _ST_NOSTATUS    => 127;
```

The **ignore calculated status changes** field defines if the action is also related to the calculated status. This option makes sense in the case of group and node entities because they have calculated status as well as regular status.

The **inherit to children** file defines if the action will be inherited to the entity's children (in the **location tree** structure). For more detail,s see the **Binding actions** section later in this chapter.

The **command** field defines which command will be used by the action.

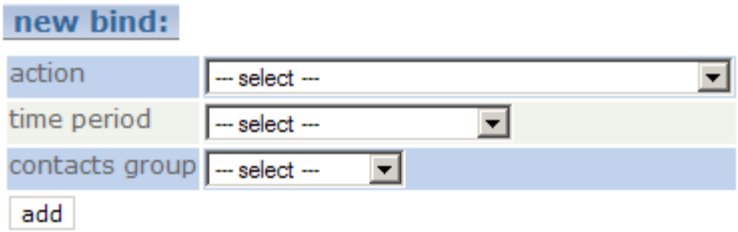
The **active** field defines if the action is enabled or not. Only enabled (the **active** option checked) actions are used. It allows for temporarily disabling the action.

Binding actions

When actions, time periods and contact groups have been created they have to be connected to applicable entities, otherwise they are ineffective. This process is called **binding**.

Actions in AKK@DA are inherited based on the **location tree** structure, but unlike contacts, only one-level-down. If you bind an action to an entity, this action affects this entity and this entity's direct children (if an action is bound to a group entity, it also works for nodes inside of this group but not for node services; if an action is bound to a node entity, it also works for this node services). This inheritance can be disabled by using the **inherit to children** option in the action configuration.

To manage bindings, in top menu, click the **actions** button and then click the **bindings** tab. To add a new command, use the **new binding** form. Select an entity to be bound. Optionally select a child entity (in that case action will be bound to the child entity instead of the parent entity). To add a new binding use the **new bind** form**.** To update or delete existing bindings, use the **update bind** form. The meaning of fields in both forms is exactly the same. All fields are mandatory.



In the **action** field choose an action to be bound to an entity. Select a time policy in the **time period** field. Select a contact group in the **contacts group** field.

A binding of a specific entity can be also viewed and managed by opening the specific entity (see the **Display entities** chapter), the **actions** tab.

8.6. Managing alarms

Overview

AKK@DA reports all detected faults, overloads, congestion and suspected results of tests by raising alarms. While displaying an entity (see chapter **Displaying entities**), related alarms are always shown on the **alarms** tab. Based on the entity which you browse, you see alarms related to this entity and all its children (based on the **location tree** structure). Also while browsing a specific view, the web GUI shows on the **alarms** tab all alarms related to all entities pertaining to this view.

AKK@DA automatically discovers and starts monitoring a vast range of services and resources. But they are monitored using default values of specific parameters and thresholds. Because of that AKK@DA often generates a lot of alarms and becomes less efficient than it could be. On a huge list of alarms the operator is not able to check quickly what is wrong. This chapter covers some of the methods of dealing with alarms. It also decribes some advanced mechanisms related to alarms.

Flaps

Flap detection is a mechanism of AKK@DA which detects situations when a specific entity keeps changing its status between OK and bad. Without this mechanism, operators could continuously be alarmed by GUI or an e-mail notification about an issue continuously affecting the same entity. In addition, each change would generate a log entry. Moreover, an operator could normally miss something because the flapping entity often has OK status. A good example is a flapping network interface or CPU with jumping utilization. AKK@DA covers these problems with the flap detection mechanism.

Flap detection is by default enabled for all entities and can be disabled by setting the **flaps_disable_monitor** parameter to 1 on a specific entity. There is no option for global disabling of flap detection.

Each entity has its own **flap monitor** field which contains 16 digits. Digits are 0 or 1. After every entity test, a new digit is added at the beginning of 16-digit chain and the last digit is removed. The new digit is 0, if the status of the tested entity hasn't changed, or 1, if it has changed. Now based on the configured flap detection policy, AKK@DA checks what is in the **flap monitor** field and decides if the flapping state will be detected or not. The **flap monitor** field value can be seen anytime via the GUI. To do it open the specific entity, click the **options** tab. There, in the **parameters** form**,** among other things, the current **flap monitor** value is shows. If the entity is not flapping, there should be a string of sixteen 0.

The flap monitor value can be reset anytime. To reset the flap monitor, open a specific entity, tab **general**. If there is at least one 1 digit in the flap monitor, there is the **clear flap** button displayed – just click it.

Sometimes, usually after complex network outages, there are many entities which have flapping state detected. To speed up the process of removing flaps, instead of the mechanism described above, the global flap clearing mechanism can alternatively be used. To clear all detected flaps, go to top menu, button **tools**, in the left panel click **entities flaps clear**. In the right panel the list of all flapping entities will appear. There click the **clear** button.

An entity which is in a flapping state leaves that state after the current entity status age exceeds 960 seconds (by default). This value can be defined only globally in the **/akkada/etc/akkada.conf** file, keys **FlapTimeSlot** and **FlapDeltaMultiplier**. It is recommended not to modify these settings, because changing them changes the way the whole flap detection mechanism works. **FlapDeltaMultiplier** is a number of digits in the **flap monitor** field (which cannot be changed).l FlapTimeSlot defines the duration (in seconds) of one time slot. AKK@DA detects the entity flaps when the number of 1 digits in the **flap monitor** field exceeds the allowed number. This number by default is 4. It is globally configured in the **/akkada/etc/akkada.conf** file, key **FlapAlarmCount**. It's not recommendede to be decreased. This value can be overwritten by configuring the **flap_alarm_count** parameter for the specific entity.

With default setting, a flapping state is detected if the entity changes the status four times in 960 seconds.

Attempts

The attempt mechanism allows for delaying the alarm. By default an alarm is raised immediately after the fault has been detected. The **attempt** mechanism can be used to change the default system behavior and raise an alarm only after the specified number of subsequent tests detected the same fault.

This can be useful e.g. when there is a jumping CPU. In a normal situation CPU jumps to high utilization for a while and there is no need to raise an alarm. But it's needed to generate an alarm if CPU highly utilizes for a longer time.

The attempt mechanism can be configured for specific entities by using **attempts_*** parameters.

attempts_max_count – mandatory; alarm will be generated after a certain number of failed tests; this should be a number greater than 0, configuring this parameter on a specific entity enables the attempt mechanism for that entity,

attempts_retry_interval – optional; seconds; overwrites default the **check_period** entity; when **attempts_max_count** is defined, after the first failed test occurrs, next test schedule can be modified by this parameter.

Let's say an entity is normally tested every 60 seconds. The **attempts_max_count** is set to 2 and **attempts_retry_interval** is set to 300. When the entity is fine, it's tested every 60 seconds. When the first test has failed, an alarm is not raised and the next test will occur after 300 seconds. If the second test will also fail, an alarm will be raised.

Thresholds

Most alarms are raised because some thresholds have been exceeded. To avoid such situations, thresholds should be adjusted to your own monitored environment. This can be done by modifying global probe configurations (see the **Services** chapter for more details regarding probe configurations) or by configuring specific entities (see **Services** and **Configuring entities** chapters for more details).

Approvals

Each alarm can be optionally approved. When an alarm was approved, information is available as to who approved the alarm, when it was done and their IP address is shown.

	start time	age of data		approval
	03/13/08 08:37:09	39s		after 01h 09m 37s by pkodzis (10.127.253.202)
	03/13/08 08:38:18	02m 18s		after 01h 08m 28s by pkodzis (10.127.253.202)
	03/13/08 08:38:18	17s		after 01h 08m 28s by pkodzis (10.127.253.202)

There are two advantages of doing that. Firstly, each approval is logged into the history log, so there is an option to check if the AKK@DA operator was watching the AKK@DA alarms and how fast their reaction was. Secondly, there is an option to hide approved alarms, which helps to see new alarms which appear. Alarms also don't emit sound alarms when they are approved.

There are a couple of ways to approve alarms.

✓ located at the top of the alarm table, approves all alarms that were previously not approved; located on the right of the specific alarm, approves only this one alarm.

approves all host alarms; this affects the host which is in the specific row of the table, **parent** column.

approves all alarms of the same type (e.g. all alarms related to hard drives) .

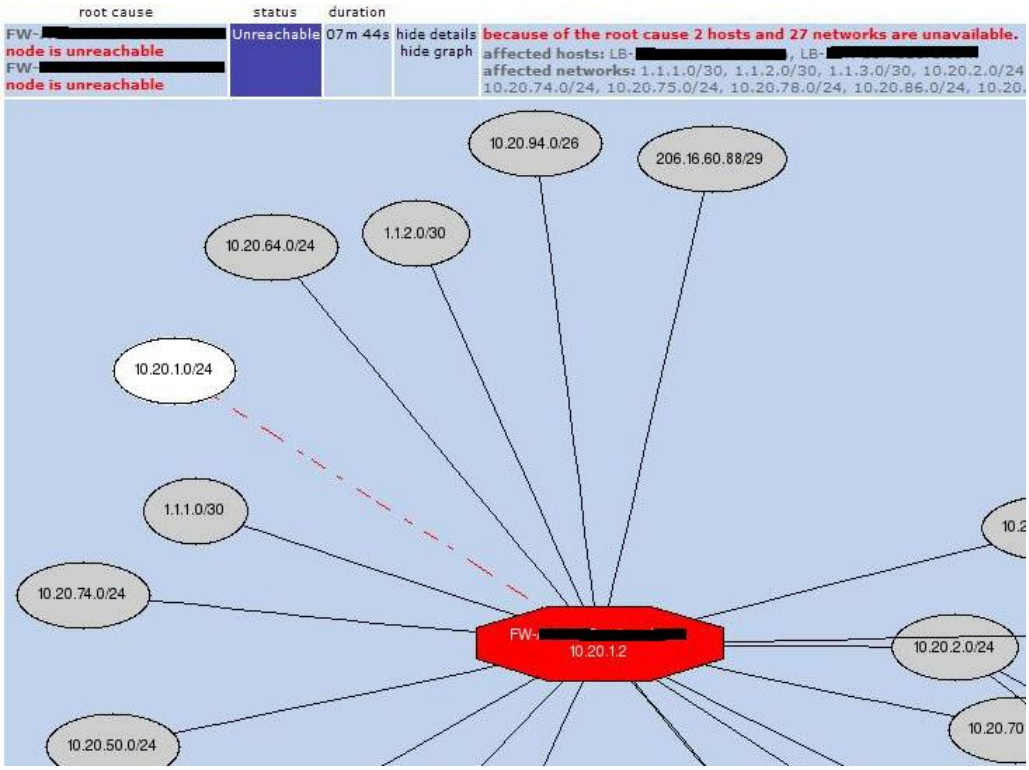
approves alarms of all members of the correlated alarm.

Correlation mechanisms

AKK@DA has two internal mechanisms which correlate detected faults. These mechanisms work all the time, but showing their results can be enabled/disabled. If needed, alarms can be browsed with or without correlation.

The first mechanism is quite simple. It correlates alarms related to monitored network interfaced cards based on their IP addresses and groups these alarms based on the IP subnet they belong to.

The second mechanism works based on the model of the monitored computer network created by the **availabe2** module (this module must be enabled instead of **available** otherwise this correlation mechanism won't work). Based on the availability of monitored network interfaces tested via ICMP it detects which connections are broken down. It also is detects which parts of the network are unavailable from AKK@DA's perspective and informs which entity is the root cause of the specific network outage.



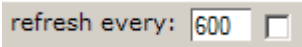
The grey color means unavailable network or device. The red color shows the root cause of the network outage. The white color means this network/device is available – it's to show the broken down connection between the unavailable part and the available network. Broken down connections are dashed red lines.

Browsing options

Hide/show approved alarms.

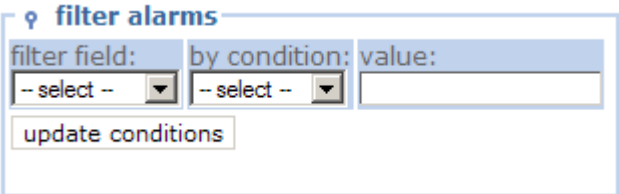
Enable/disable emitting sound in the case of alarms.

Enable/disable correlation mechanisms.



Enable/disable refreshing the AKK@DA GUI web page. The checked checkbox means enabled refreshing. The page is refreshed every 600 seconds by default. This value can be modified. After modification, the **refresh every** button should be clicked.

Filtering alarms



For filtering alarms use the form located at the bottom of the alarm table. You can set as many conditions as needed. Alarms can be filtered by error messages, entity names, parent names and status.

8.7. Managing comments

Overview

AKK@DA allows adding comments to any entity. This is a very simple mechanism, something like a very primitive blog. Entity comments are visible for all who have the right to see the entity. Also a link to entity comments is displayed with an alarm on the alarm web page if an alarm related to this entity is raised. Comments can be added on the **general** tab while browsing the specific entity. To hide/show comments use the button.When comments are enabled, at the bottom of the **general** tab, there is a form **comments** which displays comments and allows adding a new one. Each comment has its own trash icon, which deletes this specific comment. Adding and deleting comments can be limited by rights (see the **Managing security** chapter).

8.8. Managing security

Overview

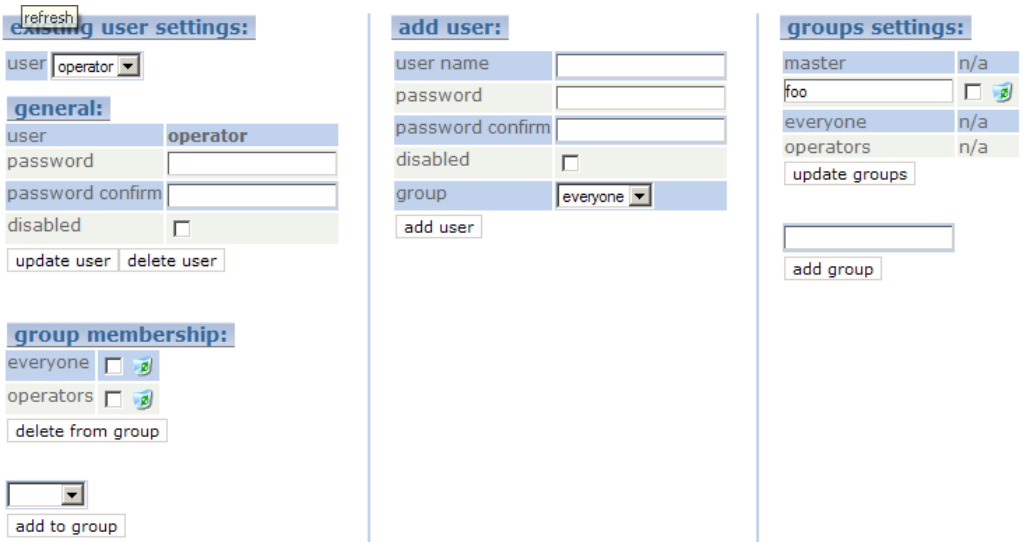
The security model used in AKK@DA is based on user group rights assigned to entities. There is no option to grant rights directly to a user. Rights are inherited based on the **location tree** structure.

AKK@DA has three built-in user groups: **master**, **operators** and **everyone**. The **master** group by default has full rights to everything. The **everyone** group by default has only view rights to all entities. The **operators** group by default has view rights to all entities and some additional rights which allow approving alarms, using utilities, making comments, etc.

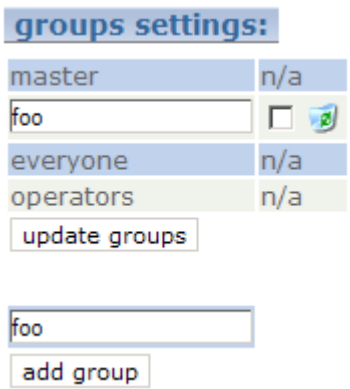
By default AKK@DA has two users: **yoda**, which is a member of the **master** group, and **operator** which is a member of the **operators** group.

Managing users and users groups

To manage users and user groups, click the **permissions** button in top menu, then click the **users & groups** tab.



To add, modify or delete a user group, use forms located in the right section. Groups have only their names. Nothing more has to be configured. Built-in groups cannot be removed.



To add a user, use the **add user** form located in the middle section. The **user name** field defines a user's ID which is used during logging into the AKK@DA process. The **password** field defines a password which is used during logging into the AKK@DA process. The passwords in both, the **password** field and the **password confirm** field must be the same.

Leave the **disabled** field unchecked to have an active user's account If this field is checked, the user's account is disabled and cannot be used to log into AKK@DA.

From the **group** pop-up menu select a group which a given user should be a member of. The user must be a member of at least one group, otherwise they have no rights to take any action.

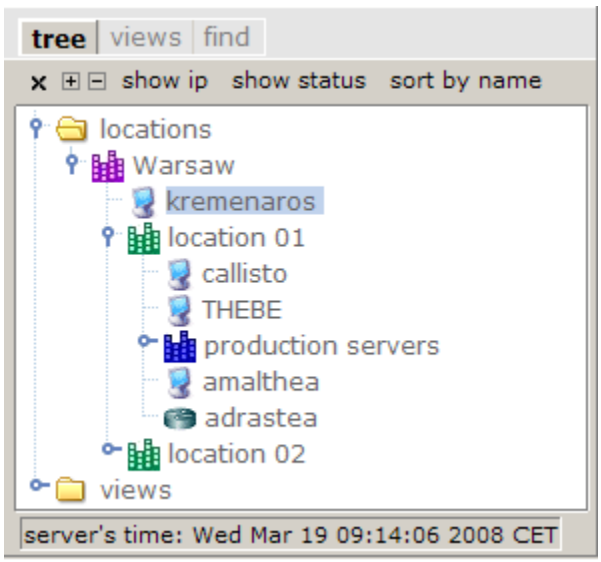
To update or delete a user, use forms located in the left section. First, select a user to be modified from the **existing user settings** form. Then the selected user can be modified or deleted by using the **general** form or its group membership can be modified by using the **group membership** form.

Groups - rights

A user defined in AKK@DA can be a member of many groups. Because all rights are assigned for user groups, not directly for users, if the user is not a member of any group, they have no rights .

If a user is a member of many groups, their rights are a simple sum of rights of all groups they belong to. E.g. if group X has the right to view everything and group Y has the right to modify everything, and a user belongs to both, group X and Y, they can view and modify everything.

AKK@DA doesn't allow using excluding rights. Groups only give rights. Groups cannot remove rights. This rule is broken when a user belongs to many user groups and their rights to a specific entity came from an inherited user group and directly set user group. In this case inherited rights are always ignored. E.g.:



A user belongs to groups X and Y. Group X has the **vi** and **vo** rights to the root object (to everything). Group Y has only the **vi** right to the THEBE node. Rights which result from belonging to group X are ignored in the case of the THEBE node, and effectively the user has only the **vi** right to the THEBE node.

Continuous

Rights to entities must be continuous to be effective for users. This is easy to explain using an example:

If a user group has the right to view the THEBE host, it must also have view rights to groups **location 01**, **Warsaw** and to the **root** object, otherwise this user group will not see the THEBE host.

The way rights are interpreted by AKK@DA has an important consequence for the way the location tree should be organized. If there is a need to grant different groups of user rights to different groups/nodes, it is recommended to keep their objects in separated groups. That makes managing rights much easier.

Types of rights

There are 8 types of rights in AKK@DA:

- **vi** (_R_VIE) – view everyone,
- **vo** (_R_VIO) – view operator,
- **co** (_R_COM) – view comments,
- **cm** (_R_CMO) – create comments,
- **ac** (_R_ACK) – approve alarm,
- **md** (_R_MDY) – modify objects,
- **cr** (_R_CRE) – create objects,
- **de** (_R_DEL) – delete objects.

The names of these rights are names only. The true information, which rights are reqiured to use a given web GUI function is defined in the **/akkada/etc/conf.d/Web/Rights.conf**. file. AKK@DA uses that file to allow or deny using every single GUI function by a user. If a user has a right defined for the specific function, it can use it. E.g.:

```
[root@sg-atx-mon5 ~]# more /akkada/etc/conf.d/Web/Rights.conf
{
      'Tools::find_mac_address' => _R_VIO,
      'Tools::cisco_locate_MAC_address' => _R_VIO,
      'Tools::entities_flaps_clear' => _R_MDY,
      'Tools::cisco_find_half_duplex_interfaces' => _R_VIO,
      'top' => _R_VIE,
      'dashboard' => _R_VIE,
      'network' => _R_VIE,
      'rights' => _R_VIE,
      'form_service_add' => _R_CRE,
```

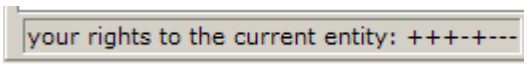
If a user has the **cr** right to the specific node entity, it can add services to this node.

After changing anything in the **/akkada/etc/conf.d/Web/Rights.conf**. file, the Apache server must be restarted to make these changes effective.

The most important right is **vi**. Without this right to the entity, a user is not able to see the entity in the GUI regardless of any other rights.

Viewing effective rights

Effective user rights to a currently selected entity are always shown in the status bar in the right panel.

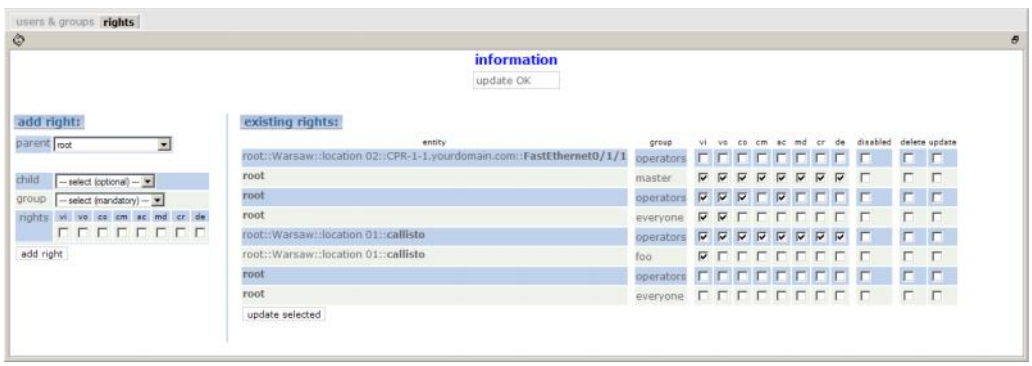


+ means user has right, **-** means user doesn't have right. **+** and **-** mean as follows: **vi**, **vo**, **co**, **cm**, **ac**, **md**, **cr**, **de**.

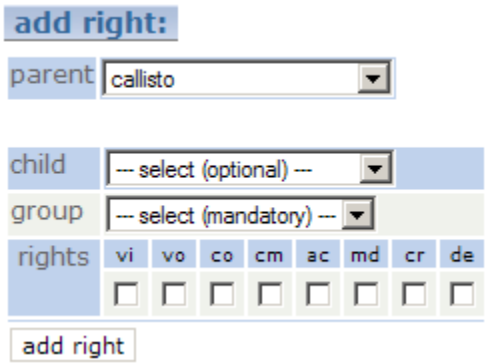
IN the right panel, on the **rights** tab detailed information is shown about user's right to the selected entity, including user groups information.

Managing rights

To manage users and user groups, click the **permissions** button in top menu, then click the rights tab.



To add rights use the form located in the left section.



From the **parent** pop-up menu select an entity which will be assigned rights. Optionally, from the **child** pop-up menu select a parent's child which will be assigned rights (in this case rights will be assigned to the child entity). From the **group** pop-up menu select a user group which willl be granted rights. In the **rights** table check rights to the selected entity granted to the selected group. Finally, click the **add right** button.

To update rights use the form located in the right section. Each record which has to be updated must have a checked checkbox in the **update** column.

8.9. Configuring entities

Overview

There are three ways of configuring entities in AKK@DA. All entity options can be configured in the right panel, the **options** tab. In the case of node entities, mandatory options of their services can be configured in one go in the right panel, in the **services options** tab. Regardless of these two ways, the entity context menu is almost always available (right click the entity to access its context menu) and some of the entity options can be modified using this context menu.

Options tab

There are four forms on the **options** tab: mandatory options, parameters, add parameter and bulk parameters modify.

The **mandatory options** form allows modifying options which belong to each entity. These options are:

mandatory options:		
name	kremenaros	
description static		
description dynamic		
check period	60	☐ with services
monitor	1	☐ with services
status weight	1	current status: OK
calculated status weight	1	current calculated status: OK
parent	Warsaw	
process		

name – is an entity name which is shown when browsing the AKK@DA GUI; it's not mandatory, but it's recommended to make it be; it doesn't have to be unique,

description static – this is an optional field, where entity description can be entered,

description dynamic – an informational field only; it shows description set by a probe process as a test result (some of probes do that – e.g. probe **nic** detects network interface descriptions defined on devices and rewrite them to the dynamic description field in AKK@DA); entity description displayed by the web GUI is a combination of the static and dynamic descriptions,

check period – in seconds; must be integer greater than 0; defines a time range between sequential tests; in the case of a node entity, there is the **with services** checkbox available – if checked, all node services will have the same value set;

monitor – must be 0 or 1; defines if an entity is enabled (1) or not (0) for monitoring; ; in the case of a node entity, there is the **with services** checkbox available – if checked, all node services will have the same value set; **warning:** disabling to monitor a node entity does not disable monitoring its services automatically – it makes the **node** probe stops monitoring the node entity, but it does not change anything regarding entity services – to disable monitoring its services check the **with service** checkbox,

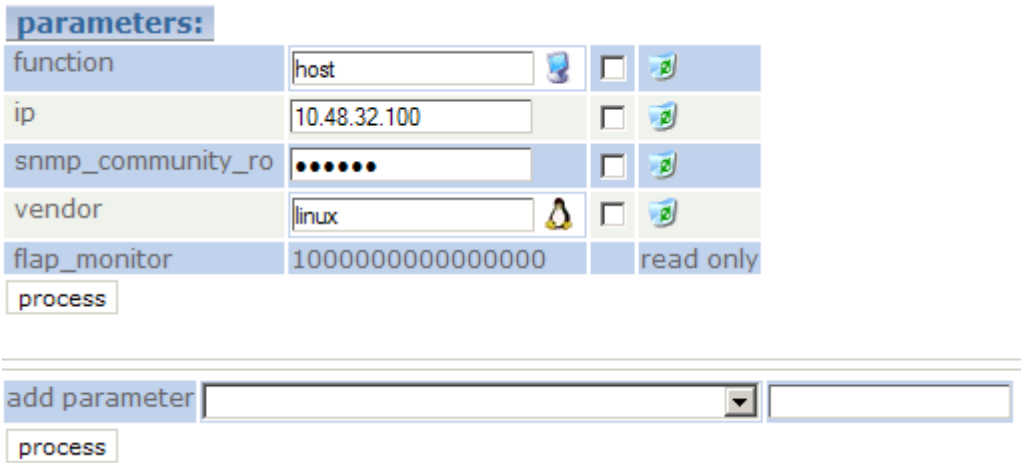
status weight – must be integer greater than -1; this value is used when entity parent calculated status is calculated; this weight can be used to make the specific entity status more/less important than the status of others; **e.g.** if a status weight is 0, it means that the entity status doesn't change its parent calculated status (it's ignored), so even if entity status is **down**, parent status is **OK**; on the other hand if a status weight is e.g. 100 it means entity parent calculated status is dominated by the status of this entity,

calculated status weight – must be integer greater than -1; this value is used when entity parent calculated status is calculated; it's used in the case of group and node entities; the idea of this weight is the same as described above in the case of **status weight**,

parent – allows managing the location tree structure; is available only in the case of node and group entities; service entities cannot be moved.

To update an entity configuration, click **process** button.

Forms **parameters** and **add parameter** are for managing entities parameters. Parameters are optional configuration fields linked to an entity if they are needed. All probe specific options, threshold options, etc. are parameters and should be configured here.



Almost all parameters are set automatically and changing their values or adding a new one is needed usually only to tune AKK@DA. Some of these parameters can be read-only.

To update parameter value, enter a needed value into the applicable field on the parameter form and click **process**.

To delete parameter, check the **trash** checkbox and click **process**.

To add parameter, select a parameter from the **add parameter** pop-up menu, enter a needed value into the field next to the pop-up menu and click **process**.

The meaning of specific parameters is described in many chapters of this user's guide. Usually they are probe type dependent – e.g for an entity which represents a monitored hard drive (is the **hdd** type), related parameters are named starting with the **hdd_** string.

The form **bulk parameters modify** allows for the modifying of a group of options/parameters in one go. It's useful when it's needed to add a lot of new nodes, or when it's needed to modify some parameters on many entities. This way of modifying is quicker but more difficult.

Data input syntax expected by this form:

```
parameter1=value1
parameter2=value2
parameter3=%%DELETE%%
…
```

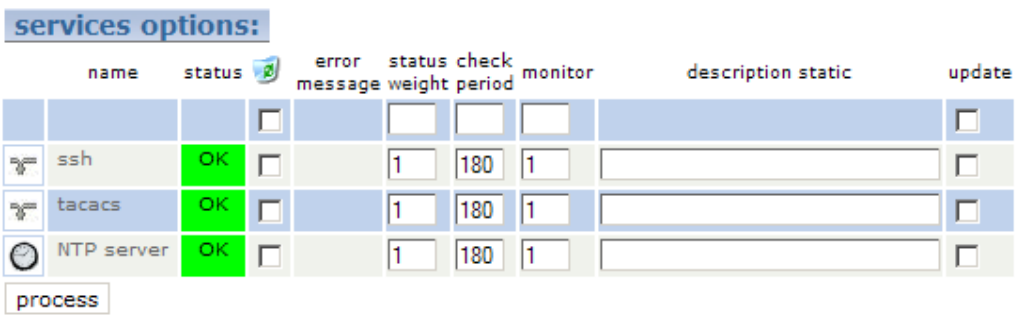
The key word %%DELETE%% deletes a parameter. Otherwise a parameter value is updated as provided. If an entity doesn't have a parameter provided, it is added.

E.g. to add a new host with IP address 1.1.1.1 and SNMP version 2 community string public, use the following string:

```
ip=1.1.1.1
snmp_community_ro=public
```

Services options tab

The form located on the service options tab allows modifying many service mandatory options in one go. The meaning of these options was described in the **Options tab** section above.



To update options, click the **process** button. Only entities with the **update** checkbox checked will be updated – if the **update** checkbox is not checked, an entity is not updated even if changes have been entered.

To delete entity, check the checkbox in the **trash** column and the **update** checkbox as well.

The first row of the **services options** table is for setting global setting. Any change in the field located in the first row will be rewritten to whole column.

8.10. Using event log

Overview

Every change of entity status is logged into the MySQL database for future use. This data is available for browsing via the web GUI at the **log** tab in the right panel. The displayed data is related only to a selected sub tree of the **location tree** or only to a related view. The displayed data can be filtered.

Browsing

Log records are shown as a table. Each record has its own unique ID (column **id**) and time stamp (column **timestamp**) which shows the date and time of events. The **name** column shows the name of an entity. Its parent entity name is in the **parent**. column The **status old** column shows the previous entity status and the **status new** column shows its current status. **Message** shows error or other message, if there is one related to the even. The icon links an event to an applicable graph – click this icon to see on the graph when the event occurred. The **approval** column shows who and when approved an alarm if it was approved.

Mon Mar 17 16:20:06 2008 - Sun Mar 16 16:20:06 2008 | > page: 1 | 2 | 3 | > | >>

id	timestamp	parent	name	status old	status new	message	approval
406171	2008-03-17 13:46:16	racks "D"	SG-ATX-[redacted]	Warning	OK		
406170	2008-03-17 13:46:15	SG-ATX-[redacted]	Kiwi CatTools	Down	OK		
406167	2008-03-17 13:34:32	SG-ATX-[redacted]	cpu	Warning	OK		
406165	2008-03-17 13:33:40	SG-ATX-[redacted]	Kiwi CatTools	OK	Down	service unavailable (stopped or not exists)	
406164	2008-03-17 13:31:25	racks "D"	SG-ATX-[redacted]	OK	Warning		

If all log records are not displayed simultaneously at the top and bottom of the log record table, there is a navigation menu for switching time ranges and pages of the log displayed. There are also three buttons at the top of the right panel available on the **log** tab:

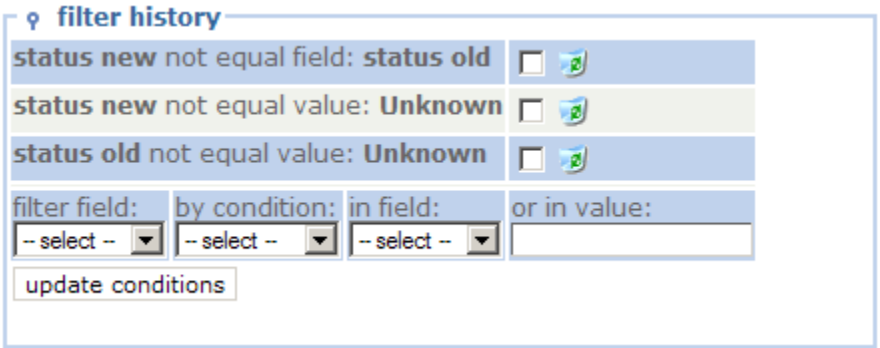
clear all log removes all records from the log; nothing stays,

keep last week removes all records older than one week from the log,

keep last month removes all records older than one month from the log.

Filtering

Log records can be filtered by using the **filter history** form located at the bottom of the right panel, on the **log** tab.



First, select the **filter field** . **Filtered field** is the column the data will be filtered by.**By condition** defines what is needed from the pop-up menu (equal, not equal, contain, etc.). If you choose any column name in the **in field** column this column will be compared with the column chosen in the **filter field** column. If, instead of choosing a column in the **in field** column, you eneter a value into the **or in value** column the entered value will be compared with the column chosen in the **filter field** column.

E.g. if the conditions are chosen as shown in the picture above records with the same old and new status, or records with the Unknown status in any of the fields will not be displayed.

To remove a condition, check the trash icon and click the **update conditions** button.

To remove all conditions, click the icon at the top of the right panel, on the **log** tab. This button is displayed only when any of the filter conditions is defined.

8.11. Using charts

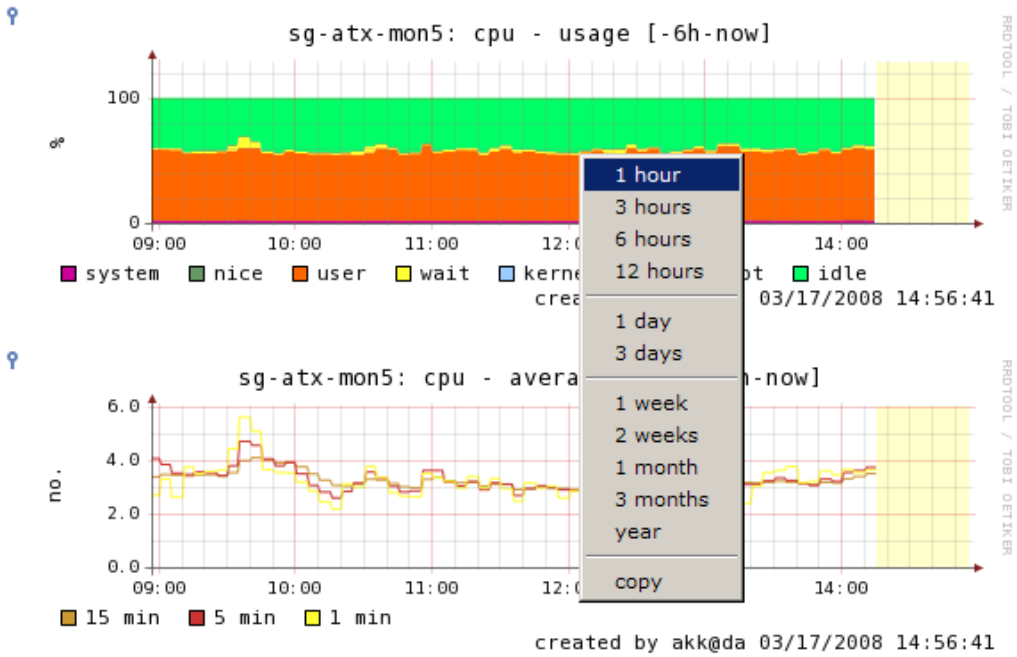
Overview

Whenever numeric data is collected during a test, AKK@DA automatically saves the numeric data into RRD databases (for more details regarding RRD databases, see the RRDTool web site at http://oss.oetiker.ch/rrdtool/). All graphs presented by the web GUI are generated using the RRD framework based on the data kept in RRD databases. The appearance of graphs is always defined by probe modules (except for the **snmp_generic** probe; in that case graph appearance is defined in template files; for more details, see the **Templates for snmp_generic probe** chapter) and cannot be changed by using the GUI interface. However, the web GUI allows changing some options such as time range, scale, size, etc.

Basic

Graphs are available while browsing a node or service entity. They are available on the **stat** tab. In the case of a node entity default charts of all of its services are shown on the **stat** tab. In the case of a service entity all service graphs are shown on the **stat** tab.

To change a presented time range, right click the graph and from the context menu select an applicable option.



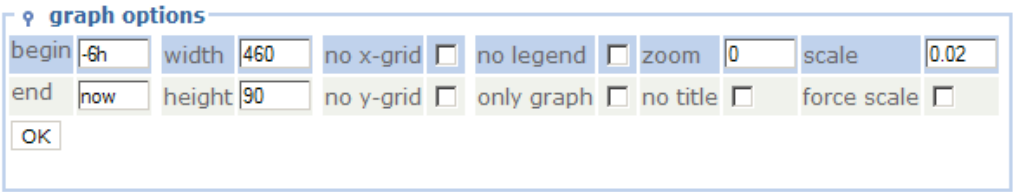
To refresh a graph, left click it once.

To get access to the regular web browser context menu, right click the graph and the context menu will appear. Then right click somewhere else on this graph and the web browser context menu will appear.

To save a graph as a file, access the web browser context menu and use the **save as**.option

Advanced

Some more advanced options regarding graphs are always available on the **stat** tab, at the bottom of the right panel in the **graph options** form.



All options in the **graph options** form are related to the RRD framework, so the best place to find more details is the RRDTool web site. If the fields are empty, default RRD values are used. To reset all options to defaults use the **reset options** button at the top of the right panel. Let's go back to the **graph options** form.

Begin and **end** defines date/time period displayed on charts. Some examples:

```
begin: end-4d, end: now – last 4 days,
begin: end-4w,  end: 00:00 – last four weeks.
```

Width/height defined chart width and height in pixels.

No x-grid disables horizontal grids.

No y-grid disables vertical grids.

No legend disables chart legend.

Only graph disables plotting of everything except for data tracks. Graphs are very raw, but still readable despite their size.

Zoom must be greater than 0. It zooms a graph by the given amount.

No title disables plotting of a title.

Scale and **force scale** are not fully tested and shouldn't be used.

Chapter 9

9. Templates for snmp_generic probe*

9.1. Available templates*

9.2. Template syntax*

9.3. How to develop new templates*

Chapter 10

10. Scripts*

10.1. Command line tools*

backup.sh

clear_all_system.sh

clear_log.sh

clear_old_sessions.pl

crypt_pass.pl

db_conn_check.pl

entitymod.pl

format_percdefl_test.pl

mib2tempraw.pl

mysql_db_dump.sh

post_install.sh

report.pl

tempraw2temp.pl

windows_service_dus.pl

10.2. Probe ucd_ext additional scripts*