

# **Notification Processor 0.24.0**

## **Technical documentation**

**Katarzyna Wladyszewska, Hadden Sp.J.**

---

## **Notification Processor 0.24.0: Technical documentation**

by Katarzyna Wladyszewska

Published April 2010

Copyright © 2003-2010 Hadden Sp.J.

HADDEN MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

All rights reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hadden Sp.J.

All trademarks included in this document are the property of their respective owners.

---

---

---

---

# Table of Contents

1. Conventions .....	1
2. General information about David system .....	2
2.1. General .....	2
2.2. David system architecture .....	3
3. Terminology .....	6
3.1. Authorization process made by David system products .....	6
3.2. David system terminology used in the documentation .....	6
4. Installation .....	7
4.1. The main configuration file of David system .....	7
4.2. Dedicated account for service of David system .....	7
4.3. Directories of David system .....	8
4.4. Configuration of syslogd daemon .....	8
5. Notification Processor requirements .....	9
6. Installation .....	10
6.1. Installation from the RPM package .....	10
6.2. Installation from the script .....	10
7. General .....	11
7.1. Functionality .....	11
7.2. Description .....	11
7.3. Related articles .....	11
8. Events Service (des) .....	13
8.1. General .....	13
8.2. Synopsis .....	13
8.3. Options .....	13
8.4. Configuration file format .....	14
8.5. Input parameters passed on running programs .....	14
8.6. Description .....	14
8.7. Related articles .....	15
9. SNMP Notification Receiver (dtrapd) .....	16
9.1. General .....	16
9.2. Synopsis .....	16
9.3. Options .....	16
9.4. Configuration file format .....	17
9.5. Input parameters passed on running programs .....	18
9.6. Description .....	18
9.7. Related articles .....	19
10. Information Recorder (dsi) .....	20
10.1. General .....	20
10.2. Synopsis .....	20
10.3. Options .....	20

10.4. Recording data format .....	22
10.5. Description .....	23
10.6. Related articles .....	24
11. Events Service Configurator (xdesc) .....	26
11.1. General .....	26
11.2. Description .....	26
11.2.1. Starting up and terminating the application .....	26
11.2.2. Main view work .....	26
11.3. Edition of list items of the main view .....	28
11.3.1. Traps range window buttons .....	28
11.3.2. The configuration of selected programs and their time ranges .....	29
11.4. Related articles .....	32
12. Buttons the most often used in Web applications .....	33
12.1. The buttons meaning .....	33
13. Trap Browser .....	35
13.1. General .....	35
13.2. Description .....	35
13.2.1. Specyfification of searching criterions .....	35
13.2.2. Generated report .....	37
13.3. Related articles .....	38

---

# List of Tables

- 1.1. The typographical conventions used in this manual ..... 1
- 2.1. David system products ..... 3
- 8.1. des options ..... 13
- 9.1. dtrapd options ..... 16
- 9.2. Arguments meaning ..... 18
- 10.1. dsi options ..... 20
- 11.1. xdesc - buttons of Edit and View menu ..... 27
- 11.2. Traps range window buttons ..... 29
- 11.3. Time ranges and programs dialog buttons ..... 30
- 12.1. The buttons the most often used in Web applications ..... 33
- 13.1. Trap Browser - a description of Traps group fields ..... 36

---

# Chapter 1. Conventions

The following typographical conventions are used in this manual:

**Table 1.1. The typographical conventions used in this manual**

Font	What the font represents	Example
<i>Italic</i>	Environment variables.	The name is kept in environmental variable <i>\$DAVIDPRIVDIR...</i>
<i>Italic</i>	Synopsis options.	<i>[-l,--log-facility log_facility]</i>
<b>Bold</b>	Names of programs and products.	<b>damcsud</b> is a part of <b>Operation Manager-a</b> .
Computer	Names of options and menus.	There is Show tool bar option in View menu.
Computer	Names of files and directories.	... reads its configuration file <code>.damadbudrc</code> .
Computer	Names of windows and dialog fields.	In A sessions property window, in Sticking string field, you can write...
Computer	Names of buttons.	Pressing Apply button lets you apply changes.
<b>Computer Bold</b>	Math formulas.	<b><math>\exp(-x)</math></b> , when $a = 0$ <b><math>1 / \text{pow}(a, a) * \text{pow}(x, a) * \exp(-x + a)</math></b> , when $a > 0$ .
<b>Computer Bold</b>	Terms used in David system terminology.	<b>SNMP Data</b> - a kind of data...
<b>Computer Bold</b>	Contents of configurations files.	<b>action</b> <b>{</b> <b>...</b> <b>}</b>

---

# Chapter 2. General information about David system

## 2.1. General

**David system** is a network management system. It is a packet of applications (modules) that allows computer network to be monitored and managed in real-time through the Internet. There is only one condition that managed devices must meet. Each device must provide SNMP (Simple Network Management Protocol) service. SNMP is the most common management protocol in the Internet so that requirement shouldn't be difficult to meet. Here is the list of typical devices that can be monitored:

- IP routers,
- ATM switches,
- manageable ethernet switches,
- UPSes with a SNMP adapter,
- TV-SAT modems that allow IP devices to work in TV cable networks,
- computers.

One of the most important feature of **David system** is its architecture. It's built of high level configurable and independent from one another modules. This principle is the most essential rule of the project. In consequences, in th metter of speaking, the same modules may build different management system. Here are the main features of **David system**:

- general thinking in information flow controlling that come form high level independence of modules of the system,
- high level configureability of the system modules that allows a special configuration of **David system** to reach end-user expectations so close as it's only possible,
- the system scalability, so you can build up the system adding additional modules in very easy way; note that these modules needn't to be part of **David system** at all; adding another monitored devices to the system is a very easy procedure,
- using shell scripts in information processing is opportunity for modeling information and influence on processing it,
- all configuration files of **David system**, files with input/output data and log files are text files,



- using SNMPv1, SNMPv2C and SNMPv3 to communicate with monitored devices.

## 2.2. David system architecture

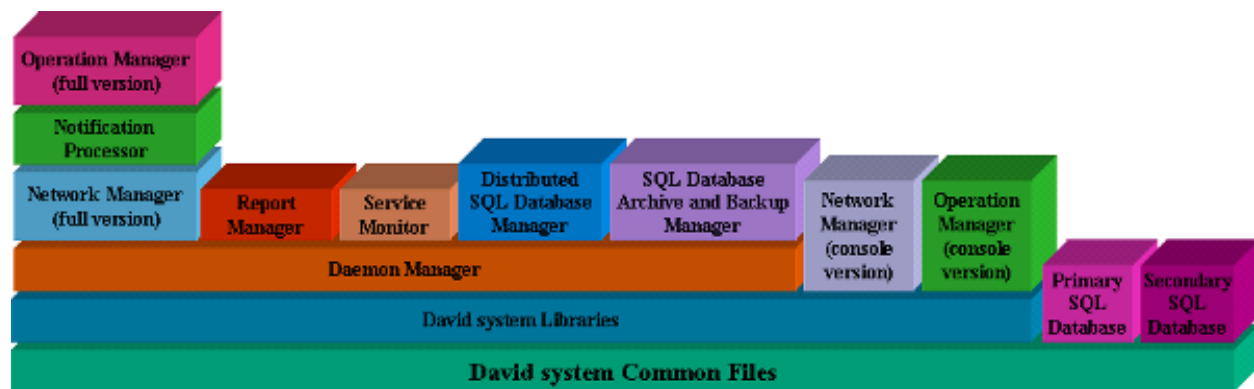
**Table 2.1. David system products**

Product	Description
David system Common Files	The product, during its installation, prepares the rudimentary directory tree for other products of <b>David system</b> . It also contains some essential and common files for all the products. Thus, this is a fundamental product of <b>David system</b> required by other its products.
Primary SQL Database	The product installs the primary SQL database of <b>David system</b> . Every single installation of <b>David system</b> must have only one the primary database.
Secondary SQL Database	The product installs the secondary SQL database of <b>David system</b> . Each installation of <b>David system</b> may have many secondary databases or none. It allows to distribute the SQL database of <b>David system</b> among many servers.
David system Libraries	This product provides libraries of <b>David system</b> required by its applications. Many other products of <b>David system</b> require that one.
Daemon Manager	It engages in running and terminating daemons of <b>David system</b> as well as monitoring of their work.
Network Manager (full version)	The product using SNMP protocol allows to visualise a topology of monitored networks and auto-discover devices in managed networks. The state of monitored devices also is visualized. The product also collects data from monitored devices using SNMP protocol and allows you to manage user accounts.
Network Manager (console version)	The product, through a graphic application, allows to visualize a topology of monitored networks and shows states of monitored resources. It allows you to control daemons monitoring devices as well as that ones gathering data. Currently, most of functions of that application is obtainable through web applications.
Notification Processor	The product chiefly engages in processing SNMP Trap notifications coming from monitored devices to management stations. The received messages can be formatted to the human readable forms, and then recorded as well. The processed notifications can be passed on to future processing.
Operation Manager (full version)	It can run specified actions on the basis of received data. Sophisticated estimation process depends on information coming from other products of <b>David system</b> and correlation of that information. It tries to build more intelligent and useful notifications then just simple reactions to incoming

## General information about David system

Product	Description
	events. The graphic application displays notifications about received events and allows to play audio files as well as reading messages by an outer speech synthesizer.
Operation Manager (console version)	The product contains a graphic application displaying notifications about events and allowing to play audio files as well as reading messages by an outer speech synthesizer.
Report Manager	The product processes recorded SNMP Trap notifications, entries about pending operations and entries about state changes of monitored devices (ping objects, network interfaces and BGP peers), and generates reports on the basis of them. Reports can be viewed using a Web application.
Service Monitor	The product monitors selected network services on application level. In order to do this it monitors selected TCP ports of specified hosts. It checks both availability of ports and a correct reaction for a few selected network protocols (HTTP, SMTP, FTP). It also can verify correctness of work of selected services by verification of received data. Results of its work can be viewed as reports and graphs made available by a Web application.
SQL Database Archive and Backup Manager	It archives the SQL Database used by <b>David system</b> applications.
Distributed SQL Database Manager	It allows to divide the database of <b>David system</b> into one primary database and many secondary ones. Such step boosts performance of the system and decreases load of the servers where daemons of <b>David system</b> work. The migration takes place during the routine work of the system. Such division may be altered many times.

Dependencies between the **David system** products are shown on the following chart..



**David system** functionality can be very large and it depends on particular configuration a lot. The most important features of **David system** are:

- discovering and visualization of monitored networks topology including visualization of states of

particular nodes;

- possibility of building control panels to monitored devices (they must support SNMP protocol), regardless of device providers;
- formatting and recording SNMP Traps sent by agents working on monitored devices;
- automatic reaction to specified SNMP Traps received from monitored devices;
- possibility of identification of an operator that has received an alert from the system about a problem;
- collecting data concerning parameters of monitored devices;
- automatic reaction to incorrect values of data that were found during data collecting;
- recording pending cases, processed by the system, which have been created as responses for events detected by the system in a monitored network;
- monitoring selected network services on application level.

---

# Chapter 3. Terminology

## 3.1. Authorization process made by David system products

The modules of David system which need to do an authorization of message senders (i.e. **damsnmpdaud**, **dnmmsd**, **dgnsd**), use the library, that checks whether an IP address of a sender matches with any record found in the file `.known.host`. The library expects to find the file in a directory pointed by a variable `confdir` in the file `/etc/system-david.conf`.

Records in the file `.known.host` are regular expressions specifying acceptable IP addresses.

## 3.2. David system terminology used in the documentation

There is an explanation of some terms, that are used in David system and its documentation:

- **messages (information)** - data received by interfaces of **Operation Manager**, its data analysers and **Cases Database Unit** of the product.
- **notifications** - the term often is used in the products: **Notification Processor**, **Operation Manager** and **Report Manager**; There are mostly data, that a source are SNMP agents working on network monitored devices.
- **events** - the term often is used in the products: **Operation Manager** and **Report Manager**; and it describes a being, that a source is SNMP Trap or SNMP Data; an **event** is always a part of a **case**;
- **cases** - the term often is used in the products: **Operation Manager** and **Report Manager**; and it describes a group of events connected one another; one **event** at last must be included in a **case**;
- **SNMP Trap** - a kind of data of **Operation Manager** product, which a source are received responses from SNMP agents; SNMP Traps aren't answers on the requests sent by a management station, but they are sent by agents managing network interfaces and processed by **Notification Processor** product;
- **SNMP Data** - a kind of data of **Operation Manager** product, which a source are received responses from SNMP agents on request which a management station sent to them by **Network Manager**.

---

# Chapter 4. Installation

## 4.1. The main configuration file of David system

The essential configuration file of David system in `/etc/david-system.conf`. It contains entries as pairs: `key = value`. Basically, except the entry `default_email_recipient`, there is no such need to modify any record in that file. All necessary modifications are made during installation processes of particular David system products. Below, there is a list of all entries along with their descriptions that may occur in this basic configuration file.

- `user` - a name of the user with which rights all daemons of David system works;
- `default_email_recipient` - the default e-mail address where messages from David system applications are sent;
- `bindir` - the directory containing David system applications (default: `/usr/bin/david-system`);
- `libdir` - the directory containing David system libraries (default: `/usr/lib/david-system`);
- `incdir` - the directory containing David system headers (default: `/usr/include/david`);
- `confdir` - the directory containing David system configuration files (default: `/etc/david-system`);
- `logdir` - the directory containing log files of David system applications (default: `/var/log/david-system`);
- `sharedir` - the directory containing various files (images, audio files, web files) of David system (default: `/usr/share/david-system`);
- `docdir` - the directory containing various files (images, audio files, web files) of David system (default: `/usr/share/david-system`);
- `vardir` - the directory containing archive files of David system SQL database (default: `/var/lib/david-system`);
- `is_sqldb_installed` - the flag that indicate whether the SQL database of David system has been installed or not.

## 4.2. Dedicated account for service of David system

There is no needs to run any David system module as superuser (usually an account `root` with UID equals 0). Even if some David system daemon requires root rights when starting, there is always possibility to specify, as one of the daemons starting arguments, a user that rights should be taken.

It is a good idea to add a new user to an operating system, under which control David system will work.

## 4.3. Directories of David system

This hierarchy depends on a particular configuration of David system. In the default system configuration, David system contains the following directories:

- `/usr/bin/david-system` - binaries and shell scripts;
- `/etc/david-system` - configuration files;
- `/usr/share/doc/david-system` - the documentation;
- `/usr/share/david-system` - graphic and audio files, web portal;
- `/usr/include/david` - David system header files;
- `/usr/lib/david-system` - David system libraries;
- `/var/log/david-system` - log files;
- `/var/lib/david-system` - archive files of the David system SQL database;

## 4.4. Configuration of syslogd daemon

David system modules use `syslog` subsystem available on UNIX platforms. Default configuration of the system modules causes that log messages are sent with `local6` facility. It may be changed for every module during its startup. Its recommended to configure `syslogd` daemon to write all messages from David system modules into one place (one or more files with characteristic name i.e.: `david.log`).

---

# Chapter 5. Notification Processor requirements

The following requirements must be met by a management platform on which **Notification Processor** will work:

- installed, compatible version of **Network Manager (full version)**.

---

# Chapter 6. Installation

## 6.1. Installation from the RPM package

You must be `root` to install the product. The typical installation looks as this one following below:

- Install the product:

```
rpm -i david-xxx-np-yyy.rpm
```

## 6.2. Installation from the script

You must be `root` to install the product. The typical installation looks as this one following below:

- Uncompress and unpack the archive:

```
gunzip david-xxx-np-yyy.i386.tar.gz  
tar xf david-xxx-np-yyy.i386.tar
```

The operations create `david-xxx-np-yyy.i386` directory in your current directory.

- Change your current directory to `david-xxx-np-yyy.i386`:

```
cd david-xxx-np-yyy.i386
```

- Read `LICENSE` file from the current directory and `CONTINUE THE INSTALLATION, ONLY WHEN YOU ACCEPT ALL CONDITIONS INCLUDED IN THE LICENSE.`
- Run the installation script:

```
./install
```



---

# Chapter 7. General

## 7.1. Functionality

**Notification Processor** makes possible:

- responding to unsolicited, SNMP-Trap messages received from monitored devices, depending on a sender and a type of the messages;
- forwarding received messages to any program (it doesn't need to be David system module) depending on a sender and type of messages (one can send an e-mail to an operator, SMS to its mobile phone etc.);
- formatting received messages to any free form;
- recording the messages;
- browsing registered messages;
- forwarding processed messages to **Operation Manager**.

## 7.2. Description

**Notification Processor** processes SNMP Trap notifications received from network nodes, that are monitored. Results of its work can influence on **Operation Manager**.

The product receives SNMP notifications from remote agents and, for each received notification, it runs specified programs and scripts depending on IP address of the message sender or type of a device on which the sender works. The running programs receive information included in messages as their invoked parameters.

**Notification Processor** also records received messages, that next can be browsed as simple reports.

## 7.3. Related articles

[Information Recorder \(dsi\)](#)

[Events Service \(des\)](#)

[SNMP Notifications Receiver \(dtrapd\)](#)

[Events Service Configurator \(xdesc\)](#)

[Trap Browser](#)

---

# Chapter 8. Events Service (des)

## 8.1. General

**des** is Event Service and it is a part of **Notification Processor**. It runs specified programs according to a value of one of received parameters: **-n**, **-p** or **-g**. **des** is dedicated to be run by **dtrapd**. In this way, information included in a SNMP-Trap notification received by **dtrapd**, is passed to **des** as its input parameters. So you can say, **dtrapd** separates stream of information taking sender IP address into consideration while **des** separates the stream with respect to a kind of messages. Note that **des** doesn't operate on a SNMP-Trap message but it only use data which are passed as its input parameters. Depending on input arguments and loaded configuration **des** runs specified programs and passes them almost all information that received itself as input parameters.

**des** reads its current configuration during startup from a file specified as one of its input parameters.

## 8.2. Synopsis

**des** can be run with the following options: **[-o]** **[-l,--log-facility log\_facility]** **[-L,--log-level log\_level]** **[-v,--version]** **[-h,--help]** **(config\_file)** **(-n event\_oid | -p specific\_type | -g generic\_type)** **[others...]**

## 8.3. Options

Table 8.1. des options

Option name	Description
<b>-o</b>	Search only first matched entry while a configuration file is processed (default: all matched lines).
<b>-l,--log-facility log_facility</b>	Choose log facility: daemon   user   local0   ...   local7 (default: local6).
<b>-L,--log-level log_level</b>	Choose log level (on stderr and syslog) i.e. messages of selected level and more important levels will be logged: emerg   alert   crit   err   warning   notice   info   debug0   ...   debug2 (default: warning).
<b>config_file</b>	A configuration filename.
<b>-n event_oid</b>	Identifier (OID) of the notification in MIB.
<b>-p specific_type</b>	Specific number of the notification.
<b>-g generic_type</b>	Generic number of the notification.
<b>others...</b>	Other arguments that are passed to run programs and not interpreted by <b>des</b> .
<b>-v,--version</b>	Display version number on stderr and exit.
<b>-h,--help</b>	Display this help and exit.

## 8.4. Configuration file format

The configuration file consists of lines. Each line of a configuration file is a single entry - a unit of information which begins from a regular expression that describes OID or a type of the notification expressed as a long integer and taken in square brackets. A generic entry looks as follows:

```
[regular expression] [day ... HH:MM:SS HH:MM:SS] ... path_to_program1
arg1 ... argN, ..., path_to_programN arg1 ... argN; [day ... HH:MM:SS
HH:MM:SS] ... path_to_program1 arg1 ... argN, ..., path_to_programN
arg1 ... argN; ...
```

Not all mentioned above items must occur in each record. A regular expression describing numbers of accepted messages is necessary. It must be taken in square brackets. A next item taken in the same brackets may occur zero or more times. Each such element describes some time range. If information comes in one of such specified time ranges, a part of the line up to the nearest semicolon or all line, if none is present, is accepted and processed. A time range must be defined as short names of days of a week (each name started by a capital) and a range of hours common for all mentioned days. If no day of a week occurs, it means that the range of hours concerns every day. If no time range occurs, it means that information must be always accepted.

A necessary item that must be specified is path to at least one program which should be run. Its arguments may follow after it. The specified programs may be more than one but they must be separated by colons.

```
path_to_program1 arg1 ... argN, ..., path_to_programN arg1 ... argN
```

A piece of information started by a possible specification of time ranges and finished with the last program and its arguments, may be only a single record in a given line and it may be ended by a semicolon (it's not necessary). If there are more such record than a obligatory one, they must be separated by colons.

## 8.5. Input parameters passed on running programs

Besides parameters which may be specified in the file, each program receives additional data which **des** receives as its input parameters. Options [-n](#), [-p](#), [-g](#) with their arguments belong to that data and all other parameters described in the [SYNOPSIS](#) as [others](#), which are not interpreted by **des**.

## 8.6. Description

**des** interprets each non-empty line of a configuration file. As the special parameter it assumes argument of [-n](#) option if it's present, or a parameter of [-p](#) option otherwise if it's present, or a parameter of [-g](#) option at least. When such special item matches to a regular expression in a given configuration line, this line is recognized as matching line and it is interpreted. A further part of the line may be divided by semicolons to records or be a single one. The last item doesn't need to be ended by a semicolon. If a current time is within in a time range of a given item, the programs specified in this record are run. **des** proceeds each

line like this until it meets the end of the file. To stop process rest of the file when the first matched line is catch, **des** must be run with -o.

## 8.7. Related articles

[SNMP Notifications Receiver \(dtrapd\)](#)

[Events Service Configurator \(xdesc\)](#)

[Information Recorder \(dsi\)](#)

---

# Chapter 9. SNMP Notification Receiver (dtrapd)

## 9.1. General

**dtrapd** is **SNMP Notification Receiver** and it is a part of **Notification Processor-a**. It is a daemon process which works all the time the system is running and it expects SNMP Traps which come through the network from remote SNMP agents. According to its configuration **dtrapd** runs for each SNMP Trap specified programs depending on a content of a message. Data included in a received message are passed on to running programs as their input parameters.

**dtrapd** reads its configuration files: `.dtrapdrc` and `.dtrapd-maybedropped-rc` during its startup. The program expects to find the configuration files in the directory `/etc/david-system`. **dtrapd** daemon won't be run if it can't read its configuration files. A way of the configuration can be modified through input parameters (options) of **dtrapd** during its startup.

## 9.2. Synopsis

**dtrapd** can be run with the following options: `[-V,--verbose]` `[-d,--dump-packets]` `[-D,--debug-snmp-lib]` `[-o,--only-first-matching]` `[-l,--log-facility log_facility]` `[-L,--log-level log_level]` `[-P,--pid-file filename]` `[-p,--port port]` `[-r,--remote-ovevent-server host]` `[-a,--address-to-accept ipaddress]` `[-A,--address-to-deny address]` `[-c,--community-to-accept community]` `[-C,--community-to-deny community]` `[-u,--run-as-user username]`  `[--extended-distribution]`  `[--load-1level double]`  `[--load-2level double]`  `[--background]` `[-v,--version]` `[-h,--help]`

## 9.3. Options

Table 9.1. dtrapd options

Option name	Description
<code>-V,--verbose</code>	Print information about each single received event on stdout.
<code>-d,--dump-packets</code>	Dump information about each single SNMP-Trap packet on stdout.
<code>-D,--debug-snmp-lib</code>	Turn on debugging of a native SNMP library, i.e.: currently library used by SNMP wrap library of David system.
<code>-o,--only-first-matching</code>	Search only first matched line while the configuration file is processed (default: all matched lines).
<code>-P,--pid-file filename</code>	Write PID to the specified file.
<code>-l,--log-facility log_facility</code>	Choose log facility: daemon   user   local0   ...   local7 (default: local6).

## SNMP Notification Receiver (dtrapd)

Option name	Description
<i>-L,--log-level log_level</i>	Choose log level (on stderr and syslog) i.e. messages of selected level and more important levels will be logged: emerg   alert   crit   err   warning   notice   info   debug0   ...   debug2 (default: warning).
<i>-p,--port port</i>	Listen directly to a specified UDP port (default: via OVEvent).
<i>-r,--remote-ovevent-server host</i>	Connect to a remote OVEvent server (default: local OVEvent).
<i>-a,--address-to-accept ipaddress</i>	Accept only messages coming with a specified IP as a source address of a message. Using this option many times allows you to build a list of accepted IP addresses.
<i>-A,--address-to-deny address</i>	Drop only messages coming with a specified IP as a source address of a message. Using this option many times allows you to build a list of unaccepted IP addresses.
<i>-c,--community-to-accept community</i>	Accept only messages coming with a specified community. Using this option many times allows you to build a list of accepted communities.
<i>-C,--community-to-deny community</i>	Drop only messages coming with a specified community. Using this option many times allows you to build a list of unaccepted communities.
<i>-u,--run-as-user username</i>	Drop root privileges and run server as the specified user.
<i>--extended-distribution</i>	Turn on extended distribution of SNMP-Trap packets (matched message number or enterprise or variables) instead of default mode (a matched sender's address).
<i>--load-1level double</i>	Drop Events matched to these ones described in <b>.dtrapd-maybedropped-rc</b> file when OS load is higher then that value and below <a href="#">--load-2level double</a> value (domyślnie: 20.0).
<i>--load-2level double</i>	Drop all Events when OS load is higher then that value (default: 30.0).
<i>--background</i>	Go to background after startup.
<i>-v,--version</i>	Display version number on stderr and exit.
<i>-h,--help</i>	Display this help and exit.

## 9.4. Configuration file format

The configuration file `.dtrapd` consist of lines. Each line of the file is a single record - information unit which begins from a regular expression describing an IP address or OID limited by square brackets. At least one white character must occur after that and next a filename of a program and its input arguments, if needed, must be specified.

### Typical lines of the configuration file

A typical line may look as follows:

```
[^10\.0\.0\.[0-9]+$] $DAVIDDIR/bin/des -o $DAVIDDIR/conf/cisco-gw.esc  
[.*] $DAVIDDIR/bin/des -o $DAVIDDIR/conf/rest.esc
```

The configuration file `.dtrapd-maybedropped-rc` is a poorer version of `.dtrapd` file. Each line includes a single regular expression describing messages, that may be dropped in the first place when a load of the operating system exceeds a specified level.

## 9.5. Input parameters passed on running programs

Besides parameters specified in the configuration file, each program receives data included in a SNMP-Trap message as its input parameters. They could be as follows:

**Table 9.2. Arguments meaning**

Argument name	Description
-s address	IP address of a sender of a message.
-a address	IP address of an agent - a sender of a message.
-c community	Community of a message.
-e enterprise	Identifier of enterprise in MIB.
-g generic_type	A generic type of a message (only in SNMPv1).
-p specific_type	A specific number of a message (only in SNMPv1).
-n notify_oid	Identifier (OID) of a message in MIB (only in SNMPv2).
-t timestamp	A timestamp of a message.

Variables included in the message are given at the end of a list of input arguments and without a preceded switch. Each variable is taken in square brackets and it has the following format: `[OID: value]`, where `OID` is a variable identifier in MIB and `value` is a value of this variable.

## 9.6. Description

When **dtrapd** daemon receives a SNMP-Trap message, it checks whether an IP address of a sender is on the list with accepted addresses, if such list exists. When the IP address is not on this list, the message is rejected. When such list is empty, `dtrapd` checks whether the IP address is on the list with unaccepted addresses. If yes, the message is rejected. Messages which are not rejected in such way, are processed in the same way regarding `community` field in the message.

For non-rejected messages specified programs are run according to the configuration of **dtrapd**. If **dtrapd** is run without [--extended-distribution](#) option and an IP address of a sender of the message matches a regular expression of a given line of the configuration file, a program specified in this line is run and data



included in the message are passed to that program as its input parameters. If **dtrapd** is run with [--extended-distribution](#) option and a number of the message or an identifier of the enterprise, or variables included in the packet match a regular expression of a given line of the configuration file, a program specified in this line is run and data included in the message are passed to that program as its input parameters.

So, if you want to run a program only for the first matched line, you must run **dtrapd** daemon with [-o](#) option.

## 9.7. Related articles

[Events Service \(des\)](#)

---

# Chapter 10. Information Recorder (dsi)

## 10.1. General

**dsi** is **Information Recorder** and it is a part of **Network Manager**. **dsi** prints data received as its input parameters in the predefined format to specified files or on stdout. Typical sources of these data are SNMP notifications received by **Notification Manager** and processed by other programs of that subsystem (i.e. [des](#) or [dtrapd](#)). Another source can be **Operation Manager**. If **dsi** is run with [-A](#) option, it asks the server of graphic notifications to display a notification for an operator. If the server can display a notification, it returns an operator identifier and its reaction time. That information is also saved. If no file is specified or the data cannot be saved in any of specified files, the output is printed on stdout. **dsi** is also designed to service (displaying and recording) cases created by **Operation Manager**.

## 10.2. Synopsis

**dsi** can be run with the following options: [[-A,--ask-server](#)] [[-l,--log-facility log\\_facility](#)] [[-L,--log-level log\\_level](#)] [[-F,--server-file socket\\_file](#)] [[-H,--server-host host](#)] [[-P,--server-host-tcp-port port](#)] [[-f,--file file](#)] [[-C,--color-foreground red green blue](#)] [[-B,--color-background red green blue](#)] [[-S,--font-size fontsize](#)] [[-i,--severity level](#)] [[-r,--display-time seconds](#)] [[-s,--source-ip IP\\_address](#)] [[-a,--agent-ip IP-address](#)] [[-c community](#)] [[-n,--number information\\_nr](#)] [[-e,--enterprise enterprise\\_oid](#)] [[-t,--time-stamp time-stamp](#)] [[-g,--generic generic\\_type](#)] [[-p,--specific specific\\_type](#)] [[-d,--message-id string](#)] [[--community community](#)] [[-m,--message message](#)] [[--ctime unsignedtype\\_number](#)] [[--mtime unsignedtype\\_number](#)] [[--active-value signedtype\\_number](#)] [[--passive-value signedtype\\_number](#)] [[--closing-reason description](#)] [[--closed-by username](#)] [[--event-ctime unsignedtype\\_number](#)] [[--event-mtime unsignedtype\\_number](#)] [[--event-id unsignedtype\\_number](#)] [[--event-hits-number unsignedtype\\_number](#)] [[--event-state \(active | passive | not-managed-here\)](#)] [[--event-msg text](#)] [[--event-successors text](#)] [[-v,--version](#)] [[-h,--help](#)]

## 10.3. Options

Table 10.1. **dsi** options

Option name	Description
<a href="#">-A,--ask-server</a>	Ask <a href="#">Graphic Notifications Server</a> for displaying a graphic notification about the received message before saving it to the files (default: don't ask the server, only write the data to the files).
<a href="#">-l,--log-facility log_facility</a>	Choose log facility: daemon   user   local0   ...   local7 (default: local6).
<a href="#">-L,--log-level log_level</a>	Choose log level (on stderr and syslog) i.e. messages of selected level and more important levels will be logged: emerg   alert   crit   err   warning   notice   info   debug0   ...   debug2 (default: warning).
<a href="#">-F,--server-file socket_file</a>	Connect with the server via a specified socket file (default:

## Information Recorder (dsi)

Option name	Description
	/tmp/dgns.ic.socket).
<i>-H,--server-host host</i>	Connect with the server on a specified host (default: connect via a socket file).
<i>-P,--server-host-tcp-port port</i>	Connect with the server on a specified host using a specified TCP port (default: 6677 but default connection is made via a socket file: /tmp/dgns.ic.socket).
<i>-f,--file file</i>	Store information in a specified file instead of stdout; using this option multiple times creates a list of files (default: only store information on stdout).
<i>-C,--color-foreground red green blue</i>	Choose foreground color (RGB format) of the message that will be displayed by the server (i.e. -C "220 0 20" means: red=220, green=0, blue=20, default: R=0, G=0, B=160).
<i>-B,--color-background red green blue</i>	Choose background color (RGB format) of the message that will be displayed by the server (i.e. -B "0 0 100" means: red=0, green=0, blue=100, default: R=0, G=220, B=0).
<i>-S,--font-size fontsize</i>	Choose font size of the message that will be displayed by the server (default: 12).
<i>-i,--severity level</i>	A severity level of this notification expressed as a double type number (default: 0.0).
<i>-r,--display-time seconds</i>	Maximum time (in sec. max: 2 <sup>32</sup> -1) this notification will be displayed by the server (default: 60 seconds).
<i>-s,--source-ip IP_address</i>	IP address of a sender of this notification.
<i>-a,--agent-ip IP-address</i>	IP address of a SNMP agent that sent this notification.
<i>-c community</i>	SNMP community of this notification.
<i>-n,--number information_nr</i>	Identifier (OID) of this notification in MIB.
<i>-e,--enterprise enterprise_oid</i>	Identifier of an enterprise in MIB.
<i>-t,--time-stamp time-stamp</i>	Timestamp of this notification.
<i>-g,--generic generic_type</i>	A generic number of this notification.
<i>-p,--specific specific_type</i>	A specific number of this notification.
<i>-d,--message-id string</i>	An unique ID string that identifies this message in <b>Operation Manager</b> .
<i>--community community</i>	The community assigned to this message.
<i>-m,--message message</i>	A human readable string that describes this message.
<i>--ctime unsignedtype_number</i>	The case creation time (in sec. since beginning of UNIX epoch).
<i>--mtime unsignedtype_number</i>	The case modification time (in sec. since beginning of UNIX epoch).

## Information Recorder (dsi)

Option name	Description
<code>--id unsignedtype_number</code>	Number of the case in a given day.
<code>--active-value signedtype_number</code>	A value which should be returned for any event marked as active by an operator during displaying of the message (default: 1).
<code>--passive-value signedtype_number</code>	A value which should be returned for any event marked as passive by an operator during displaying of the message (default: -1).
<code>--closing-reason description</code>	Description, why the case has been closed.
<code>--closed-by username</code>	A user that closed this case.
<code>--event-ctime unsignedtype_number</code>	Creation time of next event in this case (in sec. since beginning of UNIX epoch).
<code>--event-mtime unsignedtype_number</code>	Modification time of next event in this case (in sec. since beginning of UNIX epoch).
<code>--event-id unsignedtype_number</code>	An ID of next event in this case.
<code>--event-hits-number unsignedtype_number</code>	A hit number of next event in this case.
<code>--event-state (active   passive   not-managed-here)</code>	State of next event in this case (active means that something is still wrong; passive that everything is OK; not-managed-here - for your information only).
<code>--event-msg text</code>	A human readable string that describes next event in this case.
<code>--event-successors text</code>	The successors of next event in this case ( <code>\n</code> character of new line separates another successors for this event).
<code>-v,--version</code>	Display version number on stderr and exit.
<code>-h,--help</code>	Display this help and exit.

## 10.4. Recording data format

All information received as **dsi** input parameters and any data received from [Graphic Notifications Server](#) are printed as a single line of pieces of information limited by square brackets and sometimes preceded by an appropriate letter followed by the dash. A generic line is shown below:

```
-T [program calling time] -U [program calling time in UNIX format (in sec. from 01.01.1970)] -i [an argument of dsi -i option] -r [an argument of dsi -r option] -s [an argument of dsi -s option] -a [an argument of dsi -a option] -c [an argument of dsi -c option] -g [an argument of dsi -g option] -p [an argument of dsi -p option] -n [an argument of dsi -n option] -e [an argument of dsi -e option] -t [an argument of dsi -t option] -d [an argument of dsi -d option] -M [an argument of dsi --community option] -q [an argument of dsi --closing-reason option] -b [an argument of dsi --closed-by option] -m [a human readable description of the case or the notification] -u [an operator identifier received from the server] -R [notification display time received from the server] rest of data to record (other data to store),
```

aren't recognized by **dsi** as its options.

Information recorded as a value of **-m** option depends on a value of **--id** option of **dsi**. If this option isn't given (**dsi** isn't run by **Operation Manager**), an input argument of **-m** option is written. If the option **--id** is given (**dsi** is run by **Operation Manager** and the rest of information about a case and its events are given), as a value of **-m** option such information are written as: an argument of **dsi -m** option, an argument of **dsi -i** option and descriptions of each event separated by semicolons. Such description includes: state of the event, where (A) means active, (H) means passive (historical), and (NM) means not-manageable; an argument of **--event-ctime** for the event; an argument of **--event-mtime** for the event; an argument of **--event-hits-number** for the event; an argument of **--event-successors** for the event (new line characters that separates particular successors are changed into semicolons).

## 10.5. Description

The main assignment of the program is saving data in the appropriate format received as input parameters. Additional feature of **dsi** is notifying an operator about a case or a message about **dsi** is informed. To notify an operator **dsi** sends the request to [Graphic Notifications Server](#) with all information concerning the case. The server displays the notification and after that it sends the response which also contains the notification displaying time and an identifier of an operator if it's validated. The server also sends states of events accepted by an operator (an operator may change state of a event from active to passive and vice versa or leave it without changing). States of the events are always printed as the last information on stdout. Each line of that information describes a single event. The line consist of an event identifier which is a positive integer (arguments of **--event-id** option) and a number describing whether this event should be active or passive (see options: **--active-value** and **--passive-value**).

A case may include more then one event. So, all options that look like **--event-\*** (i.e.: **--event-ctime**, **--event-id**, **--event-state**) may occur more then one time. For each event all its options must occur but their order is free. Each option has an internal hit counter so you can mix the events. In this way **dsi** tries to collect a minimum quantity of whole defined events i.e. the following two events, in spite of passing them in a different order, are the same:

1)

[--event-ctime 15](#)

[--event-mtime 20](#)

[--event-id 1](#)

[--event-hits-number 2](#)

[--event-state active](#)

[--event-msg "event 1"](#)

[--event-ctime 16](#)

[--event-mtime 21](#)

[--event-id 2](#)

[--event-hits-number 3](#)

[--event-state active](#)

[--event-msg "event 2"](#)

2)

[--event-ctime 15](#)

[--event-ctime 16](#)

[--event-mtime 20](#)

[--event-mtime 21](#)

[--event-id 1](#)

[--event-id 2](#)

[--event-hits-number 2](#)

[--event-hits-number 3](#)

[--event-state active](#)

[--event-state active](#)

[--event-msg "event 1"](#)

[--event-msg "event 2"](#)

Information passed to **dsi** as its input parameters are recorded to the files (multiple using [-f](#) option creates the whole list of files). If no file is specified or data cannot be write to any file, the whole information is printed on stdout. Recording of that information is done before printing states of the events on stdout as it was said above.

## 10.6. Related articles

[SNMP Notifications Receiver \(dtrapd\)](#)

[Events Service \(des\)](#)

**Operation Manager:** Graphic Notifications Server (dgnsd)

**Operation Manager:** Graphic Notifications Presenter (xdgnp)

---

# Chapter 11. Events Service Configurator (xdesc)

## 11.1. General

**xdesc** graphic application is **Events Service Configurator** and it is a part of **Notification Processor**. It's a graphic tool which allows you to browse and edit configuration files interpreted next by [des](#). Edition procedure consists in dividing a whole record into smaller items which can be manipulated with high flexibility. The basic (indivisible) items can be edited directly. Such edition allows to keep configuration process as easy as possible.

## 11.2. Description

### 11.2.1. Starting up and terminating the application

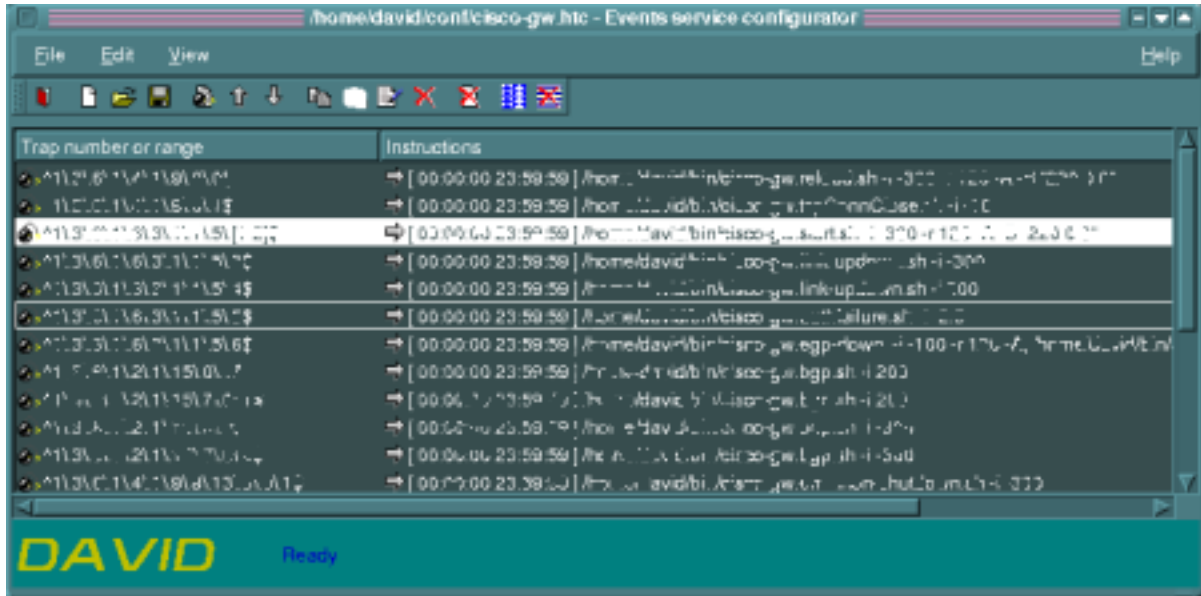
**xdesc** reads its configuration parameters from `.xdescrc` file during its startup. These parameters concern appearance of the program and other working parameters. The application expects to find its configuration file in a directory which name is kept in environmental variable `$DAVIDPRIVDIR`. When such file doesn't exist, the application begins its work with its default settings. When the application finishes its work, it records its default settings and working parameters in `.xdescrc` file. During work of the application, information about some errors can be printed on `stderr` instead of displaying of message boxes.

### 11.2.2. Main view work

Main view of the application shows the list of configuration file lines. The list consists of two columns. The first column shows a regular expression describing a range of messages. The second one shows instructions which are processed if an identifier of a given message matches to the regular expression.












#### 11.2.2.1. Main view buttons








Information presented in a main view of the application can be edited using the toolbar buttons. The first four buttons from the left correspond with options of File menu, while the next one - Edit menu. Descriptions of the buttons are showed in the chart below:

**Table 11.1. xdesc - buttons of Edit and View menu**


Button	Description
	It allows you to exit the application. Before the exit you will be asked whether you want to save changes in a modified file.
	It lets you create a new empty configuration file.
	It allows you to open an existed configuration file.
	It allows you to save an edited configuration file (when the edited file is a new one, the user is asked about its name).
Save as	There's the option of the File menu which doesn't correspond to buttons on the toolbar. It allows you to save edited document as a file which name is specified by a user.
	It allows you to add a new line of the configuration file.
	It changes an order of selected lines in the file, it moves them up.
	It changes an order of selected lines in the file, it moves them down.
	It copies selected lines.
	It pastes copied lines to the end of the list.
	It allows you to configure a selected line of the file.; it opens Traps range window.
	It deletes the selected lines.

Button	Description
	It deletes all lines displayed in the file.
	It lets you reverse selection of lines in the file.
	It allows you to unselect all lines previously selected.

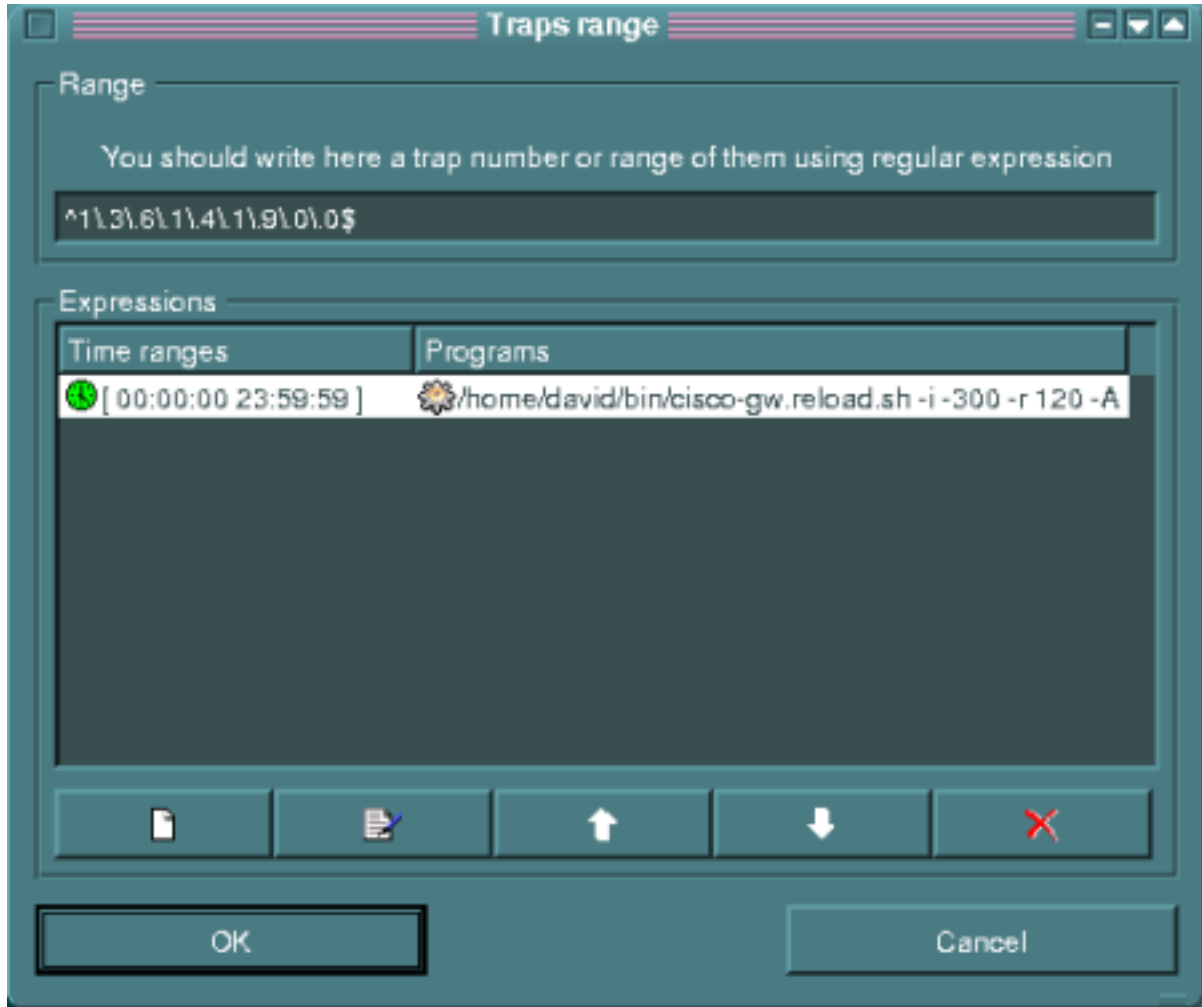
In `View` menu two options are placed additionally - `Show tool bar` and `Show status bar`. They allow you to show or hide the toolbar and the status bar.

Through `Help` menu you can find out about the application version and its creation time.






## 11.3. Edition of list items of the main view

In the main view of the application you can edit particular items of the list. Double clicking on a selected item of the list or pressing the button , opens `Traps range` dialog which allows you to configure the line. The dialog is split into two parts. The top one allows you to edit the regular expression. The bottom one consists of two columns. The first one shows the time range and the second one - programs with their arguments.


### 11.3.1. Traps range window buttons



**Table 11.2. Traps range window buttons**

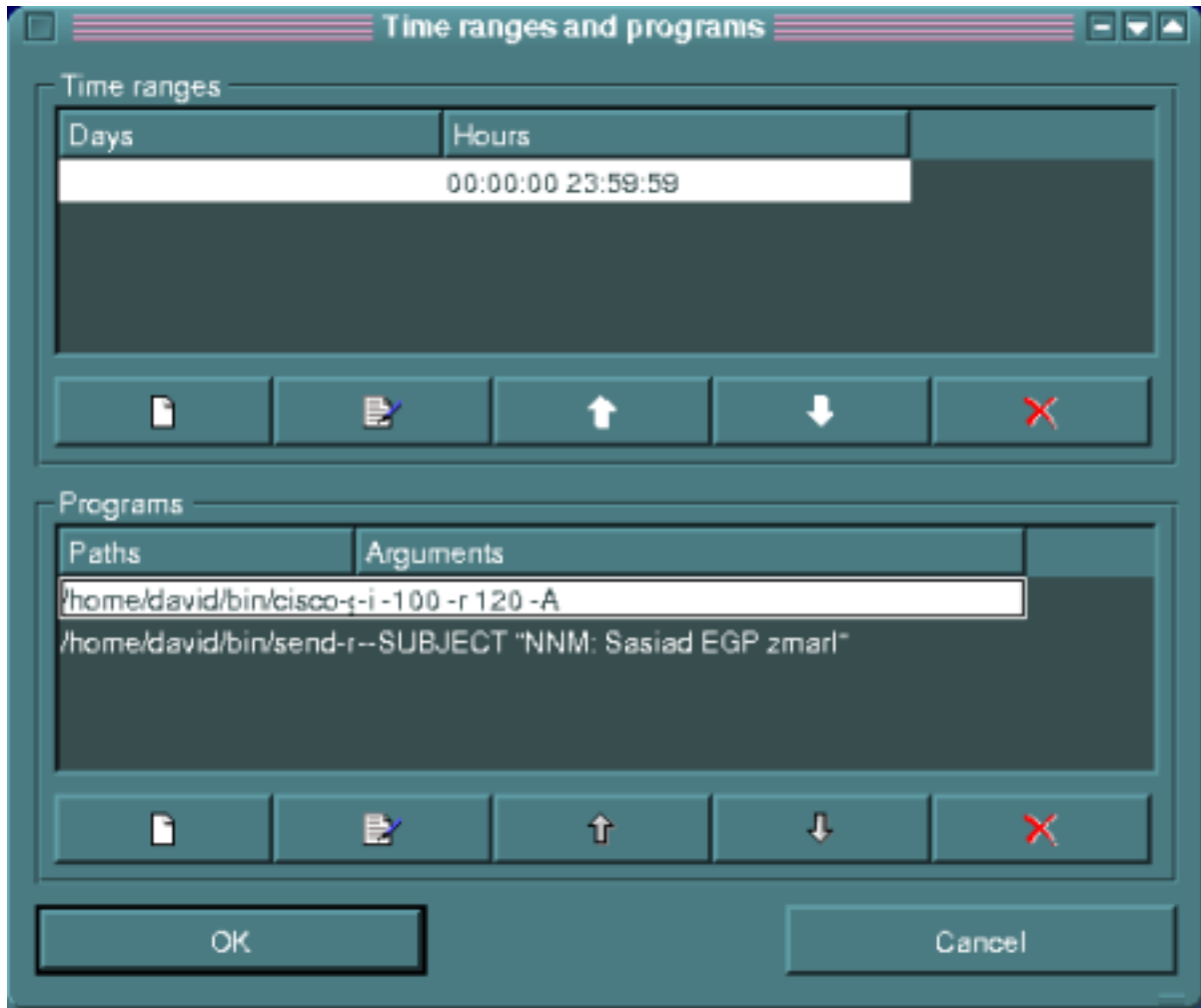
Button	Description
	It lets you add a new item to the list.
	It lets you configure a selected program or its time range.
	It changes an order of selected item; it moves it up.
	It changes an order of selected item; it moves it down.
	It lets you delete a selected item from the list.

### 11.3.2. The configuration of selected programs and their time ranges

The button  lets you configure programs and their time ranges by opening Time ranges and programs dialog. The dialog is split into two parts. Each part includes a list of items which consists of






two columns. The top list includes the days of a week and hour range. The bottom list includes paths to programs and their arguments.

### 11.3.2.1. Time ranges and programs dialog buttons

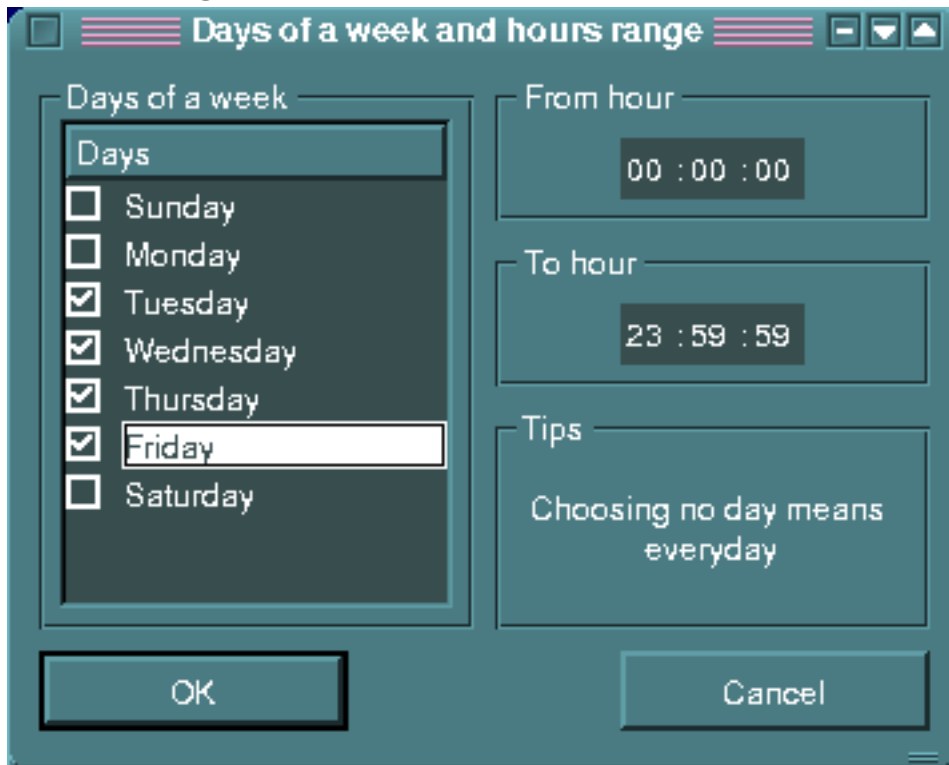


Each element of the both lists you can edit using the buttons from the chart below:

**Table 11.3. Time ranges and programs dialog buttons**

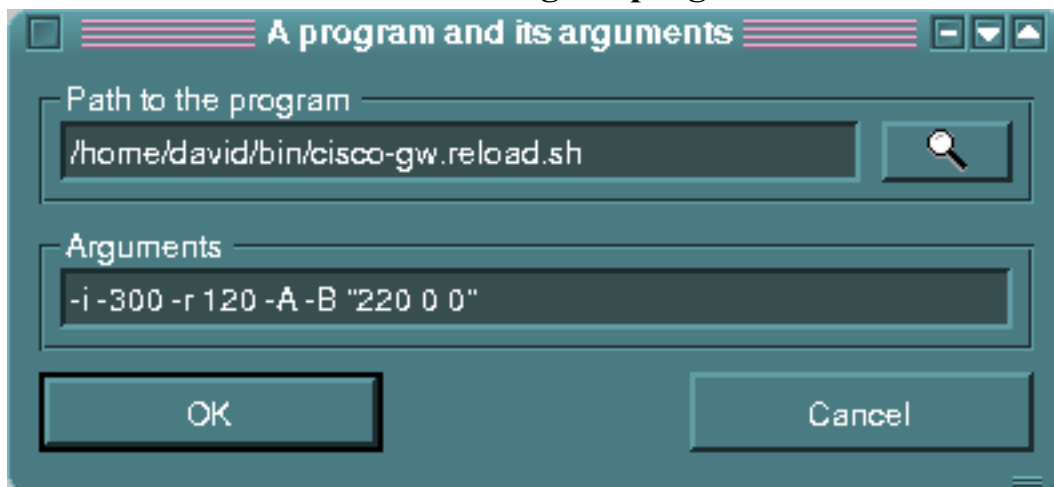
Button	Description
	It lets you add a new item.
	It allows you to edit an item describing the program.
	It changes an order of selected item; it moves it up.
	It changes an order of selected item; it moves it down.
	It lets you delete a selected item.

### 11.3.2.2. A time range edition




Choosing edition of the time range of the item opens `Days of a week and hours range` dialog, that allows you to select days of a week and hour range, when the given programs should be run. This time specification is important only when at least the hour range is valid. A lack of any time range causes the programs are run any day and any time. Non-selecting any day of a week is equivalent with selecting all of them.

### 11.3.2.3. The edition of an item describing the program



During specification of a program, that should be run, you must type a path to the program and can give

its startup arguments. The button  lets you choose the file instead of typing it directly.

## 11.4. Related articles

[Events Service \(des\)](#)











---

# Chapter 12. Buttons the most often used in Web applications

## 12.1. The buttons meaning

There are the buttons, in the chart below, that occur the most often in Web applications. Their function in particular applications is similar and even identical sometimes. Some of the buttons can have additional functions, that were described during descriptions of the particular applications.





**Table 12.1. The buttons the most often used in Web applications**

Button	Description
	It allows you to recover to a previous page.
	It deletes an item i.e.: it closes a case, sets an event in a passive state etc.
	It allows you to get to an edition of a given item.
	It confirms an operation and makes it (i.e.: generating of a report using selected criterions).
	It allows you to get to a detailed view.
	It allows you to get to a higher level of item hierarchy.
	It opens a new window with data which are prepared for a printout.
	It allows you to get to a presentation of the graph with data for a given item ( <a href="#">Collection Browser</a> ).
	It reloads a page view.
	It accepts changed values as current one.

---

## Buttons the most often used in Web applications

---

Button	Description
	It allows you to get to a report for a given item ( <a href="#">Node Reporter</a> ).
	It lets you get to a Trap browser for a given item ( <a href="#">Trap Browser</a> ).
	It lets you get to a report browser (about cases) for a given item ( <a href="#">Recorded Operation Browser</a> ).
	It saves changes, that were done by a user.



---

# Chapter 13. Trap Browser

## 13.1. General

**Trap Browser** is Web application and it is a part of **Notification Processor**. It allows to browse received SNMP Trap notifications from monitored devices.

## 13.2. Description

### 13.2.1. Specyfication of searching criterions

#### 13.2.1.1. Default view of the application



**Trap Browser** is accesible through Reports tab. It is Traps group of the tab. If you give searching criterions in Traps group, you will get a list of registered notifications.

#### 13.2.1.2. Traps group

Traps

Source IP	<input type="text"/>
Agent IP	<input type="text"/>
SNMP community	<input type="text"/>
Generic	<input type="text"/>
Specific	<input type="text"/>
OID	<input type="text"/>
Enterprise	<input type="text"/>
Text	<input type="text"/>
ID	<input type="text"/>
Community	<input type="text"/>
Source device	-- Don't mind -- <input type="button" value="v"/>
Show/hide matched	Show <input type="button" value="v"/>
Time range	Last 2 hours <input type="button" value="v"/>
Date and period	Year <input type="button" value="v"/>   Month <input type="button" value="v"/>   Day <input type="button" value="v"/>   Hour <input type="button" value="v"/>   Minute <input type="button" value="v"/>   Period <input type="button" value="v"/>
Results per page	20 items <input type="button" value="v"/>
Each bar occupies	15 minutes <input type="button" value="v"/>
Graph's width	600 pixels <input type="button" value="v"/>
Graph's height	300 pixels <input type="button" value="v"/>

Fields included in the group and their meaning are presented in the table below.

**Table 13.1. Trap Browser - a description of Traps group fields**

Field	Description
Source IP	A device IP address - of a sender of a message.
Agent IP	IP address of an agent - a sender of a message.
SNMP community	SNMP community of a message.
Generic	Generic type of a message (only in SNMPv1).
Specyfic	A specific number of a message (only in SNMPv1).
OID	Identifier (OID) of a message in MIB (only in SNMPv2).
Enterprise	Identifier of enterprise in MIB.

## Trap Browser

---

<b>Field</b>	<b>Description</b>
Text	A human readable meaning of a message.
ID	A text identifier of a message readable for a human.
Community	A message community, that is given during processing of the message (the communities are separated by ':' sign).
Source device	A device which is a sender's message.
Show/hide matched	The field means if messages, fulfilling searching criterions, are presented, or not.
Time range	Specification of time range through choosing a period of time (until now).
Date and period	Specification of time range through choosing a given date and time with a described period of time (i.e.: 1 day, 2 days, 1 week etc.).
Results per page	It describes a maximum number of entries, that will be showed at once.
Each bar occupies	It shows how much time a single bar on the graph of appearing of SNMP Traps in time will be occupy.
Graph's width	A graph width.
Graph's height	A graph height.

### 13.2.2. Generated report



The report view is divided on two parts. The top part includes a list of entries, and bottom part shows bar chart, that is a visualisation of appearing distribution of selected messages in time.

The list is divided in columns meaning a receiving time of a message, source device of a message, a message and an identifier of a message (OID). Source column instead of IP address of sender's message can include a link to **Node Browser**, that shows the sender's message. Message and OID columns includes links allowing you to go to a detailed view of a given entry. Over the row with a column description, the buttons, allowing to browse a whole list of selected SNMP Trap messages, are placed.

### 13.3. Related articles

[Events Service \(des\)](#)

[SNMP Notifications Receiver \(dtrapd\)](#)

[Information Recorder \(dsi\)](#)

**Network Manager:** Node Browser