

Trap Console™



User's Guide

September 2006



The information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictions unless otherwise noted.

Copyright (c) 2006 CSCare Inc. All rights reserved worldwide.

Trap Console is the trademark of CSCare Inc. Java and all Java based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Other product names and company names mentioned herein may be the trademarks or registered trademarks of their respective owners.

Contact: CSCare, Inc.
2880 Zanker Road, Suite 203
San Jose, CA 95134

Phone: (408) 806-6267
Fax: (408) 904-5620
E-mail: trapconsole@cscare.com
URL: <http://www.cscare.com/trapconsole>

Table of Contents

TABLE OF CONTENTS.....	3
TRAP CONSOLE USER'S GUIDE	5
OVERVIEW	5
FEATURES	6
LICENSING.....	7
REQUIREMENTS.....	7
INSTALLATION.....	8
<i>How to Obtain Trap Console</i>	8
<i>Installation with self-extracting GUI installer</i>	8
<i>Installation from RPM Package for Linux Systems</i>	11
<i>Installation from Raw Zip File</i>	11
UNINSTALLATION OF TRAP CONSOLE	12
<i>Troubleshooting</i>	12
EVALUATING TRAP CONSOLE	12
STARTING TRAP CONSOLE	13
<i>Starting Trap Console on Microsoft Windows platform</i>	13
<i>Starting Trap Console on any System</i>	14
<i>Starting Trap Console on a Port other than 6610</i>	15
<i>Starting Trap Console in Daemon Mode</i>	15
<i>Starting Trap Console to Listen for Traps on a Port other than 162</i>	15
<i>JWinSvc Wrapper Compatibility Added</i>	15
<i>Troubleshooting</i>	15
STOPPING TRAP CONSOLE	16
SUPPORT FOR SSL COMMUNICATION.....	17
<i>Prerequisites</i>	17
<i>How to Implement a Secure SSL-based Web Server</i>	17
<i>Launching Trap Console over SSL</i>	17
<i>Troubleshooting</i>	18
TRAP CONSOLE MANAGER.....	19
<i>Starting the Trap Console Manager</i>	19
<i>Logging in for the First Time</i>	19
<i>First Look at Trap Console</i>	20
<i>Changing the Password</i>	21
<i>Adding License Keys to Trap Console</i>	21
<i>Logging out from Trap Console</i>	22
WORKING WITH MIB	22
<i>Compiling a New MIB</i>	22
<i>Uploading MIB Files</i>	24
TRAPS.....	24
<i>Selecting MIB and Enterprise</i>	25
<i>Getting Trap Information</i>	26
<i>Finding out which Rules and Actions Apply to a Specific Trap</i>	26
<i>Sending an SNMP Trap</i>	27
<i>Handling Unresolved Traps</i>	28
<i>Trap Statistics</i>	28
RULES	29
<i>Reviewing Rules</i>	29
<i>Creating a New Rule</i>	30
<i>General Settings</i>	30
<i>Filtering Traps</i>	31
<i>Filtering Traps Conditionally by Expressions</i>	32
<i>Filtering Traps Based on their IP Addresses</i>	33
<i>Assigning an Action to a Rule</i>	34
<i>Limiting Rules Performing by the Time/Day</i>	34

<i>Deduplication - How to Prevent Double Processing of Identical Traps</i>	35
ACTIONS.....	38
<i>Macros</i>	38
<i>Reviewing an Action</i>	38
<i>Creating a New Action</i>	40
<i>Testing an Action</i>	52
<i>Distributing New Rules and Actions to Multiple Servers Easily</i>	54
LOG FILES.....	54
USER MANAGEMENT.....	55
LICENSING.....	56
<i>Agent Statistics</i>	57
<i>Tuning Trap Console</i>	57
<i>Working with the Application Log File</i>	59
<i>E-mail Settings</i>	59
<i>SMTP Log</i>	60
<i>Incoming Trap Restrictions</i>	61
<i>Web Browser Session Timeout</i>	62
<i>Trap Variable to Environment Variable Mapping</i>	62
<i>Environment for External Applications</i>	63
<i>Console Command Interface</i>	64
WORKING WITH DATABASES.....	64
<i>What is JDBC?</i>	65
<i>How to Write Traps to a Microsoft Access Database</i>	65
<i>How to Write Traps to an Oracle Database through JDBC-ODBC Bridge</i>	68
<i>How to Write Traps to an Oracle Database through Native Drivers</i>	70
LICENSE OVERFLOW.....	74
NEED HELP?.....	74
SNMP PRIMER.....	76
WHAT IS SNMP.....	76
THE ADVANTAGES OF SNMP.....	76
THE COMPONENTS OF SNMP.....	76
<i>The Manager</i>	76
<i>The Agent</i>	77
<i>The MIB</i>	77
HOW SNMP WORKS.....	78
SNMP TRAPS.....	78
IMPLEMENTATION.....	79
FOR FURTHER STUDY.....	79
CONTACT INFORMATION.....	80
APPENDIX.....	81
CUSTOMIZING TRAPCONSOLE.CFG.....	81
COMMAND LINE PARAMETERS.....	84
SNMP COMMAND LINE UTILITY.....	84
MACROS.....	86
<i>General Macros</i>	86
<i>Trap Message Macros</i>	86
<i>Trap PDU Macros</i>	86
<i>Trap MIB Macros</i>	87
<i>Additional Date and Time Macros for Current Time</i>	88
EXPRESSIONS.....	89
<i>Functions</i>	90
<i>Priority of Operations</i>	90
<i>Variables</i>	91
<i>Examples</i>	91
INDEX.....	93

Trap Console User's Guide

Overview

Trap Console is an SNMP trap management application, which allows users to visualize SNMP traps in a human-readable form. A user can receive pre-assigned notifications, launch 3rd party applications in response to specific SNMP traps, and forward, discard and generate SNMP traps, all that from within a simple Web browser interface.

Due to its trap forwarding capabilities, Trap Console can be used in conjunction with other network management applications to provide structured trap handling through filtering and selective forwarding of the SNMP trap traffic.

The pure Java implementation of Trap Console, together with the Web-browser user interface makes Trap Console a powerful service, that can run on virtually any hardware and/or software platform (with a Java VM available). It is accessible from any computer over Internet or Intranet via any Web browser.

The picture below shows the conceptual schema of the Trap Console operation:

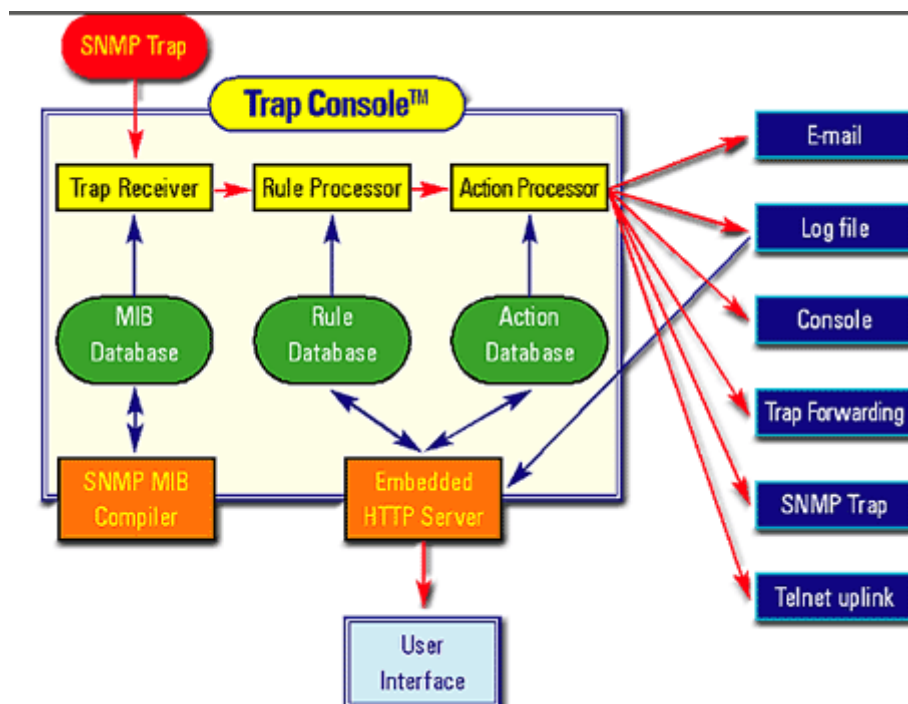


Figure 1, Trap Console Operation

Trap Console works as follows:

1. SNMP trap arrives to the Trap Receiver.
2. Using the MIB database, the Trap Receiver decodes the trap and passes it to the Rule Processor.
3. The Rule Processor matches the trap against the database of rules, pre-set by the Trap Console administrator.
4. The Rule Processor calls the Action Processor to perform actions, defined in matching rules.

Features

Trap Console, as an SNMP trap management application can:

- Receive, filter and forward SNMP traps either in SNMP v1 or SNMP v2 formats using UDP or TCP protocols.
- Handle unresolved traps (it means traps, for which the proper MIB is not available in the Trap Console MIB database).
- Integrate with higher-level network management consoles.
- Send notifications via e-mail or generic socket connections.
- Write trap information to any JDBC compliant database.
- Launch an external application with the possibility to pass trap information on a command line.
- Receive syslog messages and convert them into traps.
- Send trap information to a remote syslog server.
- Let you browse SNMP traps in the MIB database.
- Send any SNMP traps on demand.
- Perform an SNMP Set request using received trap information.
- Test a user-created action before using it in real conditions.

Due to its trap forwarding capability, Trap Console is an ideal tool for the first-stage filtering of the trap traffic. Trap Console can manage SNMP traps at remote sites, forwarding only the important traps to the high-level management console application, possibly minimizing the traffic over slower lines.

Trap Console can also communicate with other applications through its generic telnet connector module. The user-defined script controls the socket communication. Access to all properties and values of the respective SNMP trap are available with Trap Console macros.

Trap Console was designed with great care to guarantee its safe operation over long periods of time. A user can decide about how many resources to allocate for Trap Console. Trap Console makes sure it won't 'eat up' resources of the system it is running on even during temporary trap storms.

Licensing

Trap Console is licensed per number of SNMP agents it handles. If you have licensed Trap Console for 10 SNMP agents, Trap Console handles SNMP traps being received from up to 10 different SNMP agents (IP addresses). Trap Console records IP addresses from the 'agent address' field of each trap it receives. The number of unique agent IP addresses must be less than or equal to the number of Trap Console licenses (i.e. 10 in this example). All traps received from different IP addresses subsequently are indicated as the License Overflow.

In other words, should you have 10 different devices capable of sending SNMP traps in your network, you must obtain 10 SNMP agent licenses for Trap Console in order to be able to handle SNMP traps from all devices properly.

Traps over the license were just logged into the application log file and discarded in Trap Console version less than 1.3.

Starting from the version 1.3 such traps are handled as the others under the license. This should help a lot when handling traps from new added SNMP agents. This topic will be discussed later in this user's guide (see the section '[License Overflow](#)' on page 74).

Trap Console licenses come in the form of a special license key string, which you can register into running Trap Console. You can combine more license keys in Trap Console to increase handled SNMP agents.

For information on how to register a license key, see the section '[Adding License Keys to Trap Console](#)' on page 21.

Note: License keys can be registered only in the commercial version of Trap Console.

CSCare Inc. also provides an **evaluation version** of Trap Console (see [Evaluating Trap Console](#)). It works as the licensed version, but has the following built-in limitations:

- It contains only 2 licenses,
- Evaluation notices appear throughout Trap Console.

The evaluation version of Trap Console is provided only for evaluation purposes as described in 'License Agreement'.

For information how to obtain Trap Console, see the section '[How to Obtain Trap Console](#)' on page 8.

Requirements

Minimum requirements to install and use Trap Console are:

- Any operating system installed where Java 2 Runtime Environment can run,
- Minimum 60 MB free hard disk space,
- TCP/IP access to network,
- Any frame-enabled Web browser supporting JavaScript as the user interface.

Note: As Trap Console requires Java 2 Runtime Environment (JRE) to run on, the installers for MS Windows and Linux and the Linux RPM package install the necessary JRE inside the Trap Console directory.

In case that Trap Console is installed from the **zip file** distribution, JRE 1.4 or higher has to be installed prior on the computer. However, we recommend installing the latest JRE version available for your platform, which can be downloaded from <http://java.sun.com/getjava>. JRE from Sun Microsystems is recommended.

Installation

Depending on the platform, the appropriate installer should be used:

TrapConsole21Setup.exe for MS Windows systems,

trapconsole_unix_2_1.sh or trapconsole-2.1-1.i586.rpm for Linux systems.

In addition to the platform-specific installation packages, a universal raw ZIP file distribution (no setup) exists (TrapConsole21.zip), which can be used on any platform, however it needs JVM already installed.

How to Obtain Trap Console

The **evaluation version** of Trap Console installation package can be downloaded from CSCare's Trap Console download page at <http://www.cscare.com/trapconsole/download.php>

For **evaluation restrictions**, please see section **Evaluating Trap Console**.

For further information on distribution of the **commercial version** of Trap Console, please visit <http://www.cscare.com/trapconsole/sales.php> or contact us on trapconsole@cscare.com.

Installation with self-extracting GUI installer

TrapConsole21Setup.exe for MS Windows systems and trapconsole_unix_2_1.sh for Linux are platform specific GUI installers. As they are built using a single specification they behave analogously. The installation is largely self-explanatory and intuitive, so follow the setup wizard. Important steps will be illustrated using MS Windows screenshots.

The Install4j Wizard for Trap Console appears that will guide you through the process of installation. Installation process can be aborted anytime during the installation progress by pressing the **Cancel** button. Use **Next** and **Back** buttons to move between the setup screens. When the installation program successfully finishes its work, Trap Console will be running on your computer.

Selecting Destination Directory

In this step (see [Figure 2](#)), the installation program asks you for the destination folder where you wish to install Trap Console. Click the **Next** button to continue with the default folder or select a different one. If you decide to change the destination folder, select an empty one. If you intend to install Trap Console 2.1 as an upgrade of previous version, select **Migrate configuration from an existing Trap Console installation** item in the **Additional Task** screen later. It is recommended installing the application either to your local disk or to a network drive to which you have adequate access rights.

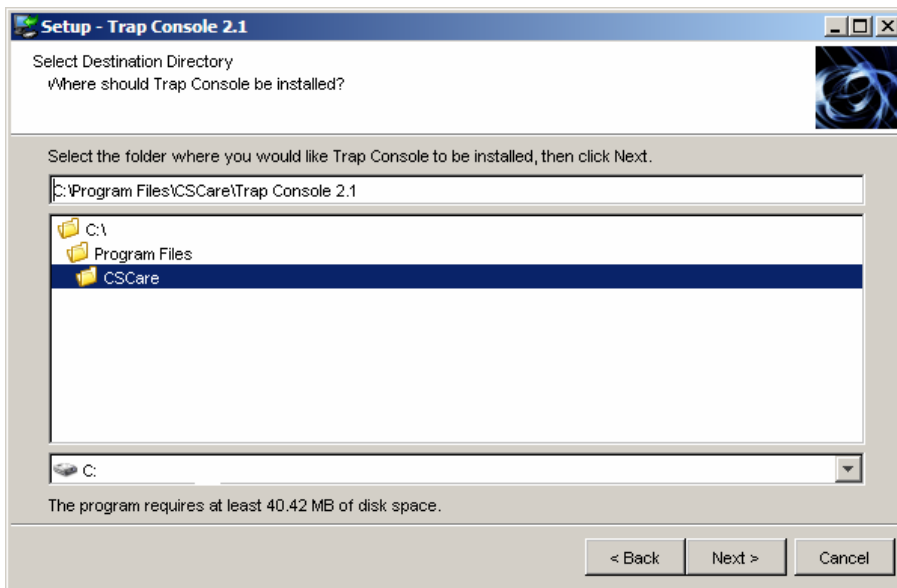


Figure 2, Choosing Destination Location

Additional Tasks

In this step (see [Figure 3](#)) you select additional tasks planned after the copying of Trap Console files. Following two tasks are available.

- Migration of previous Trap Console installation's configuration (see [Configuration Migration](#) below)
- Service installation (more in 'Installing Trap Console as a service')

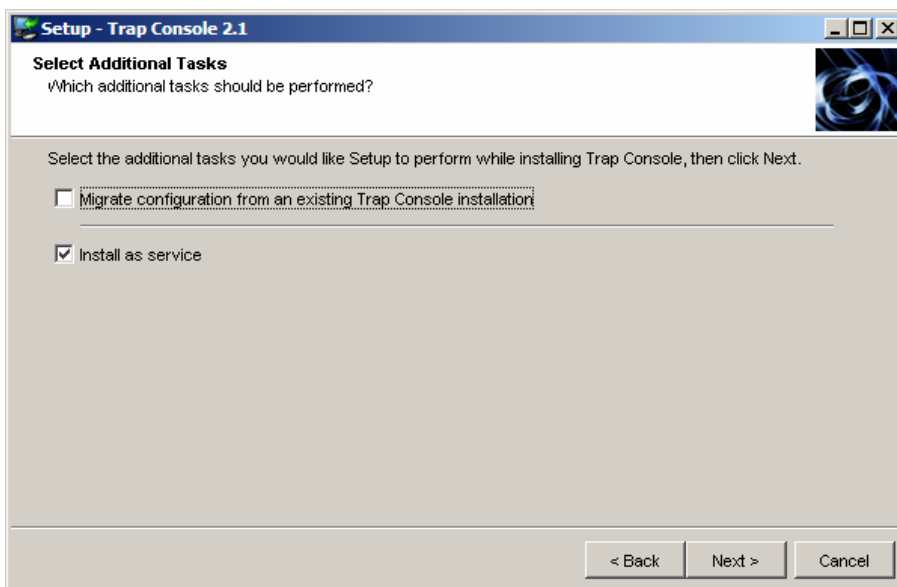


Figure 3, Selecting Additional Tasks

E-mail Settings

In this step that follows application file copying (see [Figure 4](#)) you can set global e-mail parameters.

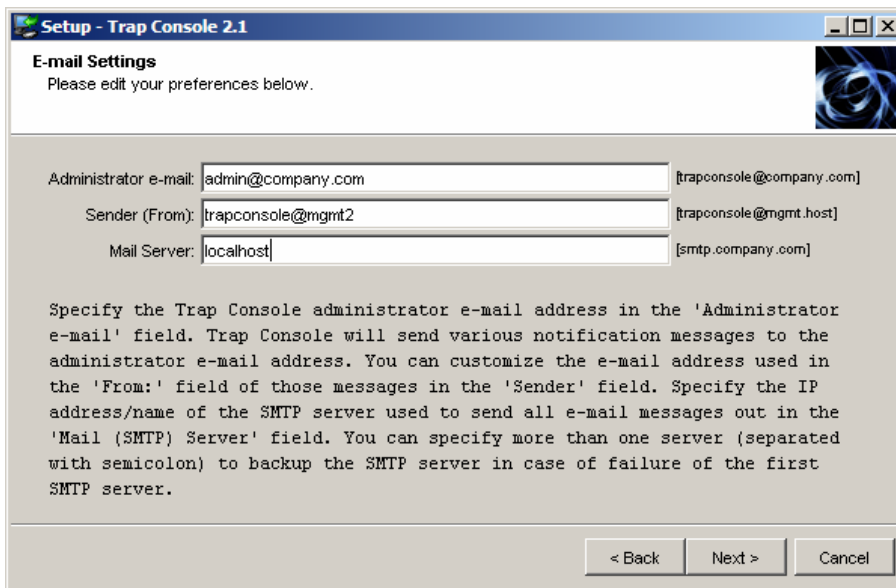


Figure 4, E-mail Settings

Trap Console can be configured to send various notification e-mail messages (see [E-mail Settings](#)). This screen expects you to specify some initial configuration values to begin with. **Administrator e-mail** is the destination e-mail address of the message. **Sender (From)** is the message originator e-mail address. **Mail (SMTP) server** is the IP address/name of the SMTP server used to send the message. E-mail messages are sent by Trap Console as a result of user defined actions or during the license overflow. In both cases the same SMTP server is used.

Configuration Migration

This optional step is activated after application file are copied, providing that it was selected on the **Additional Tasks** screen. File Open dialog is displayed, allowing you to locate the previous Trap Console installation base directory. Installer will invoke a migration utility that will update the configuration of the current Trap Console installation. Configuration Migration screen (see [Figure 5](#)) will show the output of the migration utility. You can invoke the migration utility yourself later, there is `migrate.bat` Windows batch file and `migrate.sh` shell script for Linux installed in the Trap Console directory.

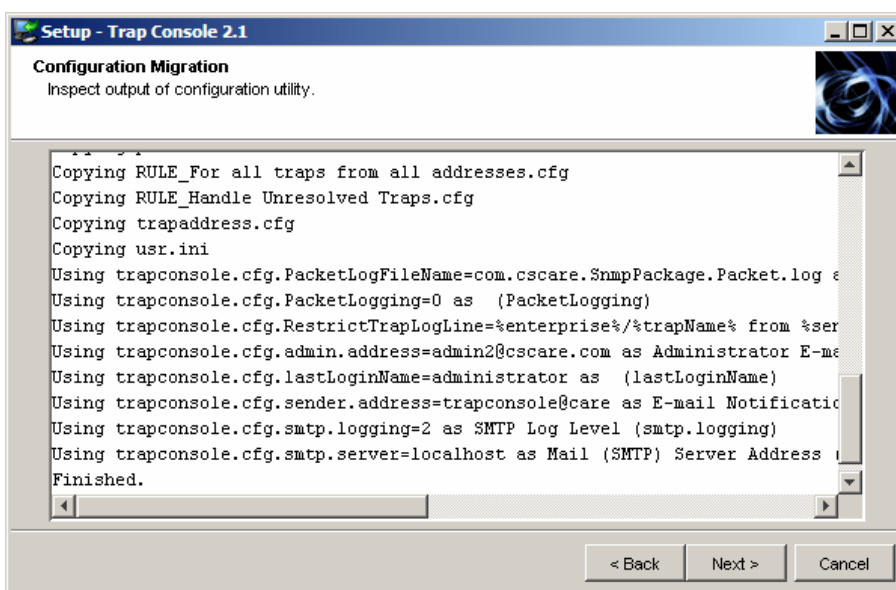


Figure 5, Configuration Migration

Installing Trap Console as a service

This optional step is activated after application file are copied, providing that it was selected on the **Additional Tasks** screen. Installation of service wrapper runs automatically, showing its output in the Service Installation screen (see [Figure 6](#)).

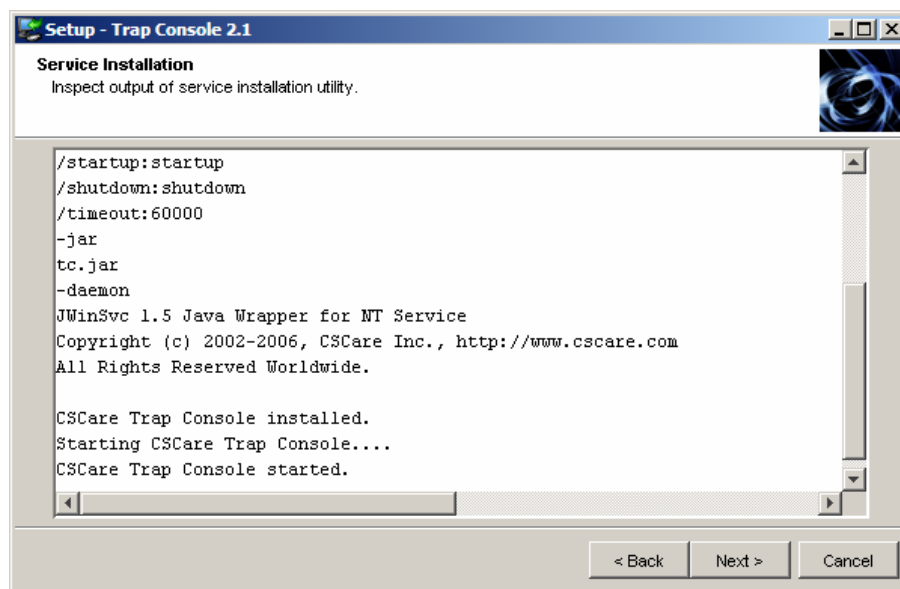


Figure 6, Service Installation Summary

Note: (MS Windows) Trap Console service can be started by selecting **Trap Console/Start Trap Console Service** from the **Start/Programs** menu. After reboot, Trap Console service starts automatically. For more detailed information how to start Trap Console Service see [Starting Trap Console on Microsoft Windows platform](#).

Installation from RPM Package for Linux Systems

Launch the `trapconsole-2.1-1.i586.rpm` file using parameter `-i` to install the package:

```
rpm -i trapconsole-2.1-1.i586.rpm
```

The rpm script installs all necessary files and Java VM to the Trap Console directory.

Installation from Raw Zip File

To install Trap Console from a raw zip file, unzip the `TrapConsole21.zip` file into a preferred directory on your system while preserving the sub-directory structure.

Launch Trap Console using the “`java -jar`” command.

Example:

```
java -jar tc.jar
```

Alternative method is starting Main class with your Java virtual machine, while adding the `tc.jar` to the CLASSPATH.

Example:

```
java -cp tc.jar TrapConsole.Main
```

or

```
java -cp /opt/trapconsole/tc.jar TrapConsole.Main -d /opt/trapconsole
```

Uninstallation of Trap Console

On **MS Windows** platforms,

- choose CSCare/Trap Console/Uninstall Trap Console from the Start/Programs menu, or
- open the Control Panel and choose the Add/Remove Programs icon. In the Add/Remove dialog (see [Figure 7](#)), select 'Trap Console' in the list of installed software and press the Change/Remove button.

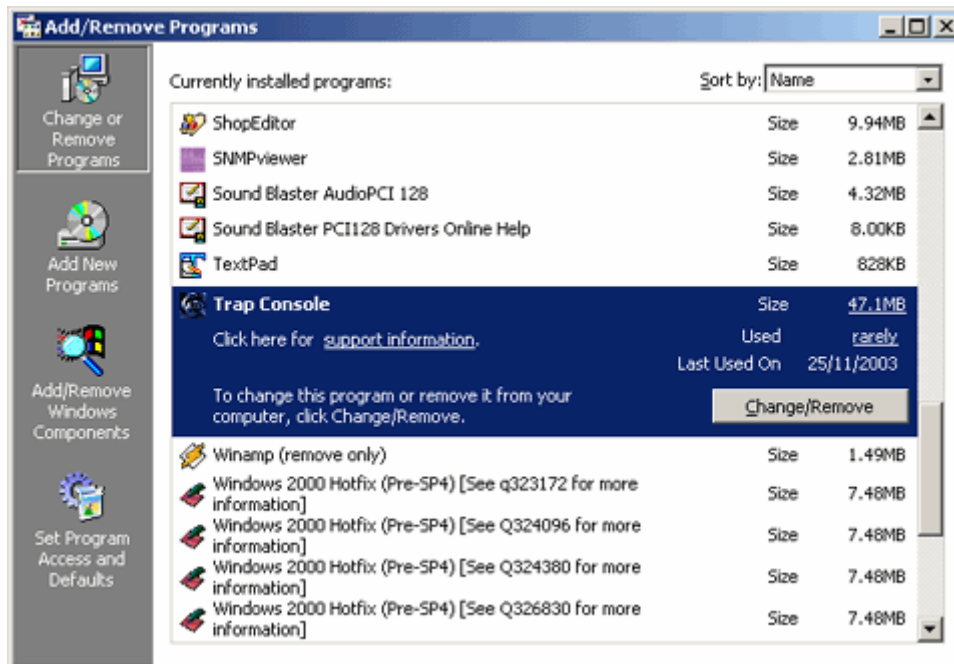


Figure 7, Uninstall Trap Console Dialog (on Windows platforms).

In the next dialog window, you will be asked for confirmation, and if you are sure you want to remove Trap Console completely from your system, click **Next** and follow the wizard.

On **Linux** systems, in case GUI installer was used to install Trap Console, invoke its uninstall script found in Trap Console directory.

For RPM package using the following command:

```
rpm -e trapconsole
```

This will remove all Trap Console folders and files except for user configuration and log files. To remove Trap Console completely, delete its entire directory tree manually.

Troubleshooting

If you have any problems with installation or uninstallation, please contact CSCare Inc. at <http://www.cscare.com/trapconsole> or via e-mail trapconsole@cscare.com.

Evaluating Trap Console

The **evaluation version** of Trap Console is provided for evaluation purposes only. It works as the fully functional licensed version in the following time restricted modes:

After installation, Trap Console works in the **unregistered evaluation mode** which expires after **7** days and Trap Console's all functions become disabled.

After expiration of the unregistered evaluation mode, registration at www.cscare.com/trapconsole/registration.php is necessary in order to receive a **registration key**, enter it in Trap Console upon login or on the **About** page and run Trap Console in **evaluation mode** further.

In the evaluation mode, Trap Console stops after 24 hours and it has to be restarted manually. Further restrictions include:

- 2 licenses are included only,
- The evaluation notices appear throughout Trap Console

Starting Trap Console

This section explains how to start Trap Console on different platforms including potential changes in the default configuration.

Trap Console installation utility creates a launch script (batch file or shell script) in its home directory. Use this file to launch Trap Console.

Starting Trap Console on Microsoft Windows platform

Trap Console can be started either as an **NT service** or in **console mode**.

Starting Trap Console as NT Service

After installation Trap Console Service is always set to 'automatic', and started.

In order to set Trap Console Service to 'manual', go to **Start/Settings/Control Panel** and open the **Administrative Tools/Services** applet. Right-click on the **CSCare Trap Console** row and select **Properties** from the pop-up menu. In the **General** tab, set the Startup type to 'manual'.

Trap Console Service can be started then from the **Start/Programs/CSCare/Trap Console** menu selecting **Start Trap Console Service**. Then if you need to access the Trap Console Manager pages in your Web browser, select **Trap Console Client** from the same menu.

Note: Trap Console Service can be started/stopped also in the above-mentioned **Administrative Tolls/Services** applet (see [Figure 8](#)).

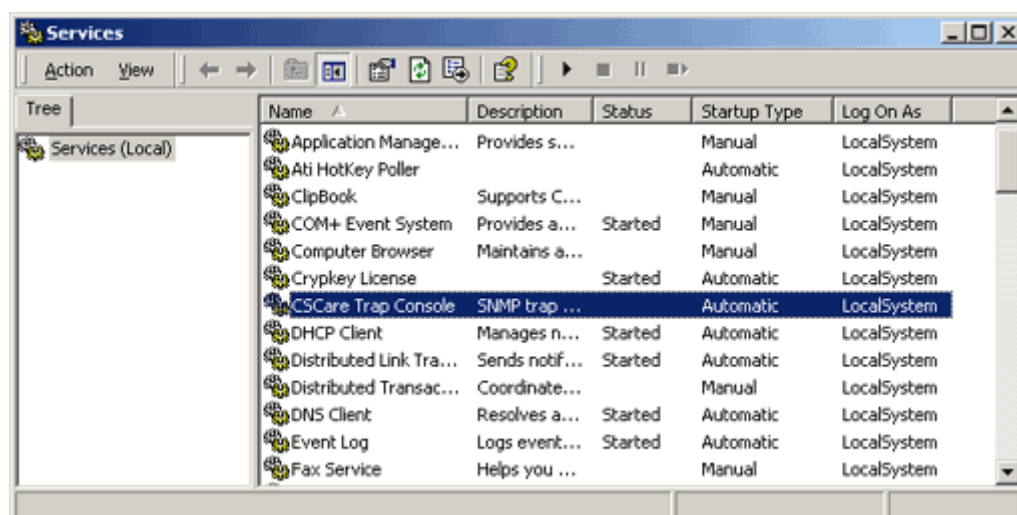


Figure 8, Services applet

Starting Trap Console in Console Mode

If running Trap Console in console mode is preferred, set the service mode to 'manual' as described above and stop it.

Then go to **Start/Programs/CSCare/Trap Console** and select **Run Trap Console in console mode**. The console window appears (see [Figure 9](#)). Then select **Trap Console Client** from the same menu to access the Trap Console Manager pages in your Web browser.

Starting Trap Console on any System

It is possible to launch Trap console following the guidelines below:

1. Include `tc.jar` file to the CLASSPATH environment variable.
2. Launch the `TrapConsole.Main` class
3. Use the `-d <home directory>` command line parameter to specify the correct working directory for Trap Console.

Examples:

```
java -jar tc.jar - applicable only for Java VM 1.2 and higher
```

```
java -cp tc.jar TrapConsole.Main
```

```
java -classpath tc.jar:/java/lib/classes.zip TrapConsole.Main
```

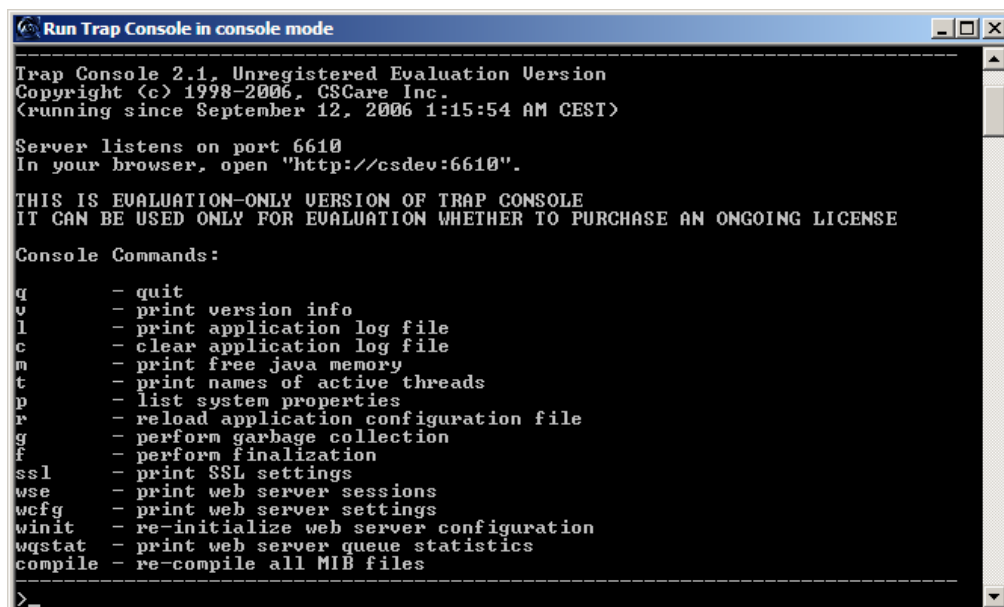
```
java -cp tc.jar TrapConsole.Main -d /home/user/trapconsole
```

The last example is a full command line used to start Trap Console with JRE on Solaris Systems.

Notice that typing the class name (`TrapConsole.Main`) is case sensitive. If you type e.g. 'trapconsole.main', the error message 'class not found' appears.

You can change the Trap Console behaviour by using command line parameters (see [Command Line Parameters](#)).

When the command line is correctly typed, the Trap Console running window appears (see [Figure 9](#)):



```
Run Trap Console in console mode
-----
Trap Console 2.1, Unregistered Evaluation Version
Copyright (c) 1998-2006, CSCare Inc.
(running since September 12, 2006 1:15:54 AM CEST)

Server listens on port 6610
In your browser, open "http://csdev:6610".

THIS IS EVALUATION-ONLY VERSION OF TRAP CONSOLE
IT CAN BE USED ONLY FOR EVALUATION WHETHER TO PURCHASE AN ONGOING LICENSE

Console Commands:
q      - quit
v      - print version info
l      - print application log file
c      - clear application log file
m      - print free java memory
t      - print names of active threads
p      - list system properties
r      - reload application configuration file
g      - perform garbage collection
f      - perform finalization
ssl    - print SSL settings
wse    - print web server sessions
wcfg   - print web server settings
winit  - re-initialize web server configuration
wgstat - print web server queue statistics
compile - re-compile all MIB files
>
```

Figure 9, Starting Trap Console

Starting Trap Console on a Port other than 6610

Trap Console's Web server listens on the TCP port 6610 by default. To start Trap Console on another port, use one of the following methods:

1. Modify the command line used to start Trap Console. Add the command line parameter `-p <port number>` in order to start Trap Console on a specific port. Example of the command line that starts Trap Console on the HTTP port 80:

For Java VM 1.2 and higher

```
java -jar tc.jar -p 80
```

For Java VM older than 1.2

```
java -cp tc.jar TrapConsole.Main -p 80
```

2. Edit the `trapconsole.cfg` file located in the Trap Console home directory. Enter the line

```
HttpServerPort=<port number>
```

e.g.

```
HttpServerPort=80
```

The command line above will make Trap Console listen on the HTTP port 80.

Note: Trap Console must be restarted for this change to take effect.

Starting Trap Console in Daemon Mode

Trap Console contains a console window, where information about the application is available to the user. The information includes number of running threads, amount of free memory, etc. However, in order to run Trap Console as a background process, it may be desirable not to display the console screen at all. You can do so by appending the command line argument `-daemon` at startup:

For Java VM 1.2 and higher

```
java -jar tc.jar -daemon
```

For Java VM older than 1.2

```
java -cp tc.jar TrapConsole.Main -daemon
```

Starting Trap Console to Listen for Traps on a Port other than 162

Trap Console listens for SNMP traps on the default UDP trap port 162. To let Trap Console listen for SNMP traps on another port select **Runtime** from the main menu and choose the tab for the appropriate trap receiver (**UDP/TCP Trap Receiver**). Enter new port number in the **Receiver Port** edit line and click **Set**. Receiver will restart if already running. No application restart is needed.

JWinSvc Wrapper Compatibility Added

Trap Console is compatible with JWinSvc Wrapper from:

<http://www.cscare.com/jwinsvc>

Note that JWinSvc has been tested and works. This is a viable option for running Trap Console in the NT service mode under Windows NT 4.0, 2000 and XP.

Troubleshooting

If you have any problems with starting Trap Console, please look at the Trap Console's console window. There you can find information about errors occurred during the Trap Console start (i.e. busy TCP port 6610, busy UDP port 162 etc.).

Note: There is no user readable output from Trap Console if it works as an NT service. All application system output is redirected to the application log.

To see those events, open **Start/Settings/Control Panel** and find the **Administrative Tools** icon. Then choose the **Event Viewer**. Select the **Log** menu in the Application log file. Look for events having Trap Console displayed in the Source field. In this case one event in the Event Viewer corresponds to one line in the Trap Console's console window.

Stopping Trap Console

Trap Console can be stopped as follows:

- From its Java console, typing **q** and pressing 'Enter'
- From Web browser, accessing the **Runtime/Console** page, typing **q** in the provided edit line and pressing **Submit**.

Note: This option is available only for the administrator.

- If Trap Console works as NT service, it can be stopped from the **Start/Programs/CSCare/Trap Console** menu choosing **Stop Trap Console Service**.
- Another way how to stop Trap Console NT service is to open the **Control Panel** and stop it in the **Services** applet (see [Figure 8](#)) right-clicking on the Trap Console row in the list and choosing **stop** from the pop-up menu, or pressing the **Stop** button on the main toolbar.
- Trap Console can be stopped using the command line switch

```
-stop [<timeout-seconds>]
```

where timeout is in seconds and is optional. If the timeout is not defined, the default value (5 seconds) is used.

Let's presume that Trap Console is running. It can be stopped by starting a new instance with parameter `-stop`, e.g.

```
java -jar tc.jar -stop 2
```

where the already running instance of Trap Console will be stopped in 2 seconds provided the both instances were started from the same working directory.

- If Trap Console is installed as system daemon on Linux with System V init (default with RPM package and optional with other installers) it can be stopped using script installed in `/etc/init.d/trapconsole`. Invoke it directly

```
sh /etc/init.d/trapconsole stop
```

or using your distribution's service utility e.g.

```
service trapconsole stop
```

Support for SSL Communication

Trap Console supports SSL communication on server's HTTPS port. By default, the HTTPS port 6610 is used. Command line parameter `-ssl` can be used to launch Trap Console over SSL.

Prerequisites

SSL-enabled applications must run on Java 2 platform. However, there are several versions of Java 2 platform that slightly differ in terms of offering packages for SSL communication support.

In order to run applications supporting SSL communication, 'Java Secure Socket Extension' (**JSSE**) is needed. Java Runtime Environment (JRE) 1.4 contains JSSE.

To run an SSL-based Web server, the server SSL **keystore** has to be generated. SSL keystore is generated with '**keytool.exe**' which is located as follows:

- in `<javahome>\bin` for **JRE** platforms
- in `<javahome>\bin` and `<javahome>\jre\bin` for **JDK** platforms

How to Implement a Secure SSL-based Web Server

The Java runtime platform which is going to be used to implement a secure SSL-based Web Server must contain JSSE. If the Java runtime platform does not contain JSSE, it must be installed.

Generating SSL Keystore for Server

SSL communication is based on the public key cryptography. The SSL server needs to have its own private and public keys generated.

Keystore is a binary file which contains keys and certificates. Access is password protected. For more information regarding keystore and key management, please visit the keytool documentation pages at:

<http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html>

SSL **keystore** for the server is generated using the 'keytool' application (**keytool.exe**).

- For **Microsoft Windows** operating systems, Trap Console root folder contains a prepared batch file named **genSSLKey.bat** which can launch 'keytool.exe'.
- For **Linux/Unix** operating systems, Trap Console root folder contains a shell script named **genSSLKey.sh** which can launch 'keytool.exe'.

Please make sure that `<javahome>\bin` is in the path, so that 'keytool.exe' can be started comfortably.

After launching 'keytool.exe', a keystore is generated according to the options pre-defined in the batch file / shell script. The keystore is then placed in the Trap Console working directory.

The `-dname` option in the batch file / shell script may be edited if necessary, to personalize the 'keystore' file. For more information visit the keytool documentation pages mentioned above.

Signing the Keystore

In order to implement a completely valid SSL server, the key has to be signed by a valid and trusted Certificate Authority (e.g. Verisign, Entrust, etc). More details on this topic can be found on the keytool documentation pages at: <http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html>

Launching Trap Console over SSL

Launching Trap Console over SSL correctly, the following two files must be accessible:

- **'keystore'** – this file is generated as described above, and is located in the Trap Console current working directory. Its default password is *ssl/Password*.
- **'cacerts'** – A file used to keep the root certificates of signing authorities. Its default password is *changeit*. It is shipped with Java platform and is typically stored in:

<javahome>\lib\ext\security for JRE, and

<javahome>\jre\lib\ext\security for JDK

'cacerts' file is stored in PKCS #12 format containing public but no private keys. It may also contain SSL keys. Since all the Certificate Authorities in the cacerts file are implicitly trusted for code signing and verification you must manage the cacerts file carefully. The cacerts file should contain only certificates of the Certificate Authorities you trust.

If these two files are located elsewhere, or their passwords have been changed, the application must be told where to find them and what are their correct passwords, otherwise the application will behave incorrectly. To avoid any strange behaviour, the exact location and the password can be specified using the following command line options typed as Java commands in the command line while launching Trap Console:

- `-Djavax.net.ssl.keyStore=keystore_location` – specifies the path to the 'keystore' file
- `-Djavax.net.ssl.keyStorePassword=keystore_password` – tells the application the 'keystore' password
- `-Djavax.net.ssl.trustStore=truststore_location` – defines the path to the 'cacerts' file
- `-Djavax.net.ssl.trustStorePassword=trustword` – tells the application the 'cacerts' password

Troubleshooting

In case of having difficulties launching or running Trap Console over SSL correctly, the detailed debugging trace message can be obtained specifying the following Java command option while launching the application:

```
-Djavax.net.debug=all
```

This option will print the detailed step-by-step debugging message on the screen, which can be used to trace the possible errors or incorrect behaviours.

Trap Console Manager

Once you have Trap Console installed properly it is ready to manage SNMP traps.

Trap Console is a system application running in the background. The main goal of the Trap Console service is that it is written in Java programming language, thus it works both on UNIX and Microsoft Windows systems.

Trap Console works with two kinds of user data:

1. The first of them are various port numbers, queue lengths etc. Trap Console reads this data at its start from the `trapconsole.cfg` file, which is stored in simply editable plain text form. All necessary information on how to change the Trap Console system setting can be found either in the [Customizing trapconsole.cfg](#) section of the Appendix in this user's guide or in the `readme.txt` file. Data stored in the `trapconsole.cfg` file can be suppressed by command line parameters (see [Command Line Parameters](#)).
2. Other user data are various Actions, Traps and MIB definitions stored in appropriate (same named) files. They are managed via the Trap Console Manager. Their meanings and ways of changing them are the topic of our interest in this chapter.

Starting the Trap Console Manager

In order to start working with the Trap Console Manager, there are a few options:

- choose **CSCare/Trap Console/Trap Console Client** from the **Start/Programs** menu,
- start the `Default.html` page located in the Trap Console directory,
- or alternatively, start running your Web browser and type URL of the **host** and **port**, where Trap Console was started, or

Typical examples of **URLs** that you might use are:

```
http://server:6610
```

(connects to Trap Console running on its default port 6610)

or

```
http://server
```

(connects to Trap Console, provided it was started on the HTTP port 80)

where the word **server** is replaced with an **IP address** or a **domain name**.

To start working with your local installation of the Trap Console Manager type the URL address as

```
http://127.0.0.1:6610
```

Note: The Trap Console Manager consists of HTML pages located in the `TrapConsole\Templates` folder. Every page contains a short help to the displayed topic in the **Description** section.

Logging in for the First Time

As it has been introduced to you earlier, Trap Console contains an embedded Web server. It is possible to access Trap Console from any place, where a Web browser is available, no matter whether it is inside your Intranet or over the Internet. Password protection, together with advanced session management guarantees a secure use of Trap Console.

At the Trap Console Manager start, the **Login** dialog shown in the [Figure 10](#) appears.

Welcome to Trap Console. Enter your login name and the system password.

Login Name:

Password:

(Use empty password for the first login)

Figure 10, Trap Console Login

Here you should type in your **Login Name** and **Password**. By default, 'administrator' is set as the login name. For the first login, use the 'administrator' login name and leave the Password edit line empty. Once you have logged into the Trap Console Manager, you will be allowed to change the Password as you need (see the subsection 'Changing the Password' on page 21).

To log in under a different login name than 'administrator', clear the Login Name edit line, and type in the login name you wish to use.

If you type the wrong password, the following message appears:

Welcome to Trap Console. Enter your login name and the system password.

Invalid password

Login Name:

Password:

(Use empty password for the first login)

Figure 11, Login Failure

Do not worry about it and try to retype the correct password (delete the input box if the password is blank). The number of attempts is not limited.

First Look at Trap Console

At first sight you should notice the hypertext links styled like tabs (see Figure 12) at the top of the page:

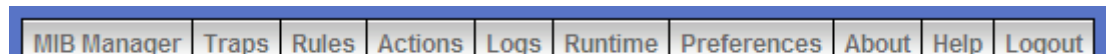


Figure 12, Trap Console Navigation Bar

This menu bar allows you to invoke the desired actions. By clicking an appropriate item you can work with

- **MIB manager** (including MIB files),
- **Traps**,
- **Actions** to be performed according to previously defined **Rules**,
- Action **Logs** files,
- **Runtime** control and statistics
- Trap Console **Preferences**,

- **Help**, and
- **Logout** Trap Console.

Note: The page invoked by the item **Preferences** is displayed as default - i.e. you see it immediately after logging, without clicking the Preferences hypertext item (or if you click it, then nothing happens). Other pages are invoked by clicking the appropriate hypertext item.

Changing the Password

In the section 'Logging in for the First Time' you have been introduced to how to log in Trap Console. Once you are logged in you might wish to customize your Trap Console. To change your password, go to the **Preferences** page, select the **Password** link from the tabs at the top of the Preferences frame and the following page appears (see [Figure 13](#)):

The screenshot shows the 'Preferences' page with a tabbed interface. The 'Password' tab is selected. Below the tabs, there is a 'Change Password' link. A message states: 'Currently logged in user: administrator. You can change the password of the currently logged in user here.' Below this message, there is a form with the following elements:

- A 'User:' dropdown menu with 'administrator' selected.
- A 'New Password:' text input field.
- A 'Retype New Password:' text input field.
- A blue 'Change Password' button.

Figure 13, Change Password Dialog

Select the user, type the new password and retype the new password. Then click the **Change password** button. If your new password has been retyped correctly, the Change Password Result is OK.

If not, then repeat typing in the new password.

Adding License Keys to Trap Console

Once you have purchased a License Key (see the section 'Licensing' on page 7), you will need to register it in Trap Console.

The Trap Console license key can be entered in the Web browser.

Note: The below described license key manager is available only in the **commercial version** of Trap Console.

In the Web browser, go to the **Preferences** page. Under the **Licensing** tab, there is a **License Key Manager** section (see [Figure 14](#)).

Preferences

Licensing	Application Log	Address Restrictions	Restricted Trap Log	Mapping	Variables	E-mail
-----------	-----------------	----------------------	---------------------	---------	-----------	--------

Trap Console is licensed per number of unique SNMP agents (measured by unique source and agent addresses within

License: 5 SNMP sender or agent addresses
Trap Console operates within limits of the license.

[SNMP Agent Statistics](#)

[License Key Manager](#)

Trap Console's license keys. For security reasons, serial numbers instead of actual license keys are being displayed.

Please, contact <http://www.cscare.com/trapconsole> to obtain licenses.

New License Key:

Key Serial Number	Number Of Licenses	
778	5	<input type="button" value="Remove"/>
Total: 5		

Figure 14, License Key Manager

Type your License Key into the **New License Key** input box and click the **Add** button. If the License Key is valid, it appears in the list of registered keys. In case of error message, try to retype your key or contact CSCare Inc. at <http://www.cscare.com/trapconsole> or via e-mail on trapconsole@cscare.com.

Logging out from Trap Console

To stop working with Trap Console Web browser Manager you should log out. To do so, simply click the **Logout** item. You will be logged out immediately. If you want to continue working with Trap Console Manager you must log in again. Thus, as a result of Logout the **Login** page appears on the screen. If you want to finish your work, simply leave this page.

Note: Logging out of Web browser Manager has no influence on whether Trap Console runs or stops. In spite of logging out of the Web browser Manager the status (running or stopping) of the Trap Console Service remains unchanged. To stop Trap Console see the section 'Stopping Trap Console' on page 16).

Working with MIB

MIB is an information base needed to resolve traps (see 'SNMP Primer' on page 76). A trap arriving to Trap Console is resolved only if it is 'known' to MIB. Fortunately, the trap which is not found in the MIB can be still processed by Trap Console as an 'unresolved' trap.

Trap Console is equipped with some of the most common MIBs. They are located in the `mibs` folder both in the source and compiled form.

Trap Console allows you to use any other (your own) MIB. Using the Web Manager you can view and even compile its source and append to used MIBs.

Compiling a New MIB

To compile a new MIB file, place it to the `mibs` folder, which is considered by Trap Console as a MIBs 'home' folder. Then select the **MIB Manager** page (see Figure 15) in your Web browser.



Figure 15, MIB Manager Page

All MIB files present in the MIB 'home' folder (i.e. mibs) appear in the bottom frame. MIB files not yet compiled have undefined MIB Name (indicated by '-not compiled yet-' note there)

To compile a MIB, select its link in the list of MIB files. The selected MIB file appears above the list. Finally, press the **Compile** button and the MIB Manager issues a message (see Figure 16) as to whether the compilation was successful or not at the end of the compilation..

To compile all MIBs, press the **Compile All** button.

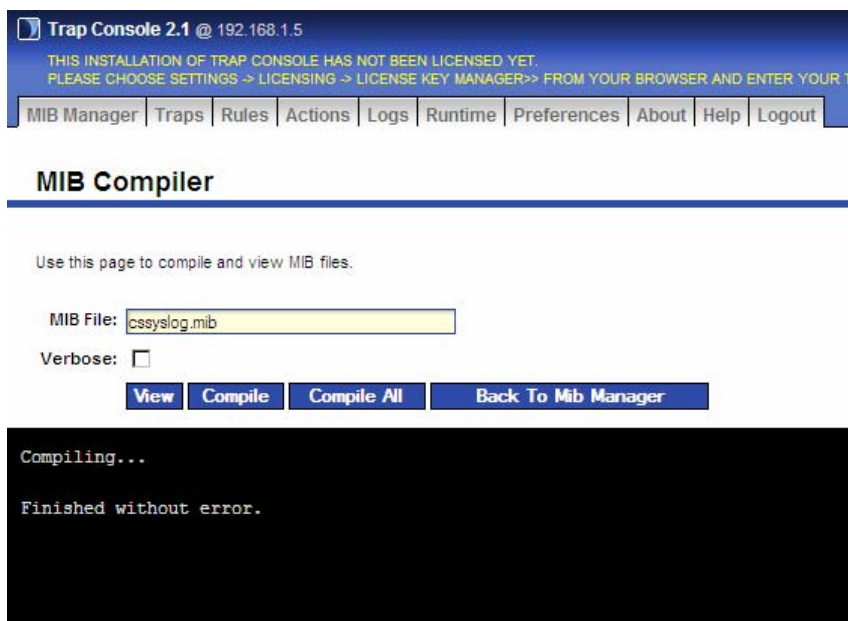


Figure 16, The MIB is compiled

Click the **View** button to view the MIB's source, or click the **Back To Mib Manager** button to return back.

Note: To view a detailed compilation report, tick the **Verbose** check box and press **Compile** again.

Uploading MIB Files

New MIBs can be uploaded directly from the Web browser.

Click the **Upload MIB File** button (see [Figure 15](#)) to upload a new file. You can, of course, use any other way to copy the desired file into the `mibs` subdirectory under the Trap Console home directory. The page shown in [Figure 17](#) appears:



Figure 17, MIB File Upload

- Enter the MIB file name into the **File Name** edit line, or press the **Browse...** button to select the mib file name.
- Click the **Perform Upload** button.

Trap Console copies the selected file into its `mibs` subdirectory.

Traps

Starting from the version 1.3., Trap Console supports both SNMP v1 and SNMP v2 encoded traps. In previous versions, SNMP v1 was supported only.

This paragraph describes how Trap Console can handle SNMP traps. It can:

- Receive all incoming traps
- Send any SNMP trap on demand
You choose a trap and send it to a selected address (see the section '[Sending an SNMP Trap](#)' on page 27).
- Send an SNMP trap as a result of a user-created action
You create an action, which sends a selected trap to a selected host address (see the section '[Sending Another SNMP Trap](#)' on page 46).
Note: If you send an SNMP trap, you can choose the appropriate encoding version.
- Forward an SNMP trap as a result of a user-defined action
This action takes a received trap and forwards it to a selected host (see the section '[Forwarding a Trap to Another Host](#)' on page 45).
Note: In this case you have several possibilities of trap encoding:
Outgoing trap has the same encoding as the received one (by default).
Trap Console encodes a trap according your wish as SNMP v1 or SNMP v2.

On the **Traps** page you can (see [Figure 18](#)):

- get information about what traps can be processed by Trap Console with respect to the used MIB file and the set Enterprise,
- see trap information,
- see rules defined for chosen (highlighted) trap, and
- send a trap.

SNMP Traps

This page displays information about all SNMP traps currently known to Trap Console. Traps are grouped by MIB files and by trap enterprises.

Trap: enter the trap name/object ID:

 or select below:

MIB:

Enterprise:

Traps:

- csCare Trap
- snmp.linkUp
- snmp.coldStart
- snmp.linkDown
- snmp.authenticationFailure
- egpNeighborLoss
- snmp.warmStart

Trap: linkDown
Enterprise: snmp.2
OID: 1.3.6.1.2.1.11.2
Description: A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent
Severity: unknown
ifIndex [OID: 1.3.6.1.2.1.2.2.1.1] INTEGER. A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value remain constant at least from one re-initialization of the entity's network management system to the next re- initialization.

Figure 18, SNMP Traps Page

In this section we focus on common questions concerning working with traps.

To obtain information about a particular trap, enter the trap name or Trap Object Identifier (object ID) if known into the first edit line (see Figure 18), or select a trap from the given hierarchical lists.

Selecting MIB and Enterprise

The **MIB** list box in Figure 18 allows you to select the active MIB file. It can look as follows:

MIB:

- all --
- CS-WATCH-MIB (mibs\cswatch.mib)
- CSCARE-PING-MIB (mibs\cspingtrap.mib)
- CSCARE-SYSLOG-MIB (mibs\cssyslog.mib)
- CSCARE-TRAP-MIB (mibs\cscaretrap.mib)
- IF-MIB (mibs\IF-MIB.my)
- RFC-1215 (mibs\rfc1215.mib)
- SNMPv2-MIB (mibs\SNMPv2-MIB.my)

Figure 19, MIB List

where items in the list mean that either **all** MIBs or the named MIB are selected.

Note: The MIB listed in the MIBs list must be present in the `mibs` folder.

The **Enterprise** list box allows you to select an Enterprise

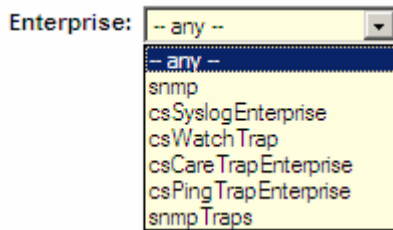


Figure 20, Enterprise List

where items in the list mean that either **any** Enterprise or the named Enterprise is selected.

Getting Trap Information

To get information on a selected trap, simply click the **Show trap information** button. The available information including the **Trap** name, **Enterprise**, **Trap Description** and **Severity** appear in the lower part of the table (see Figure 18). Furthermore, **OID** (Object Identifier), the unique identification of the trap and **Trap Values** if any are displayed as well.

For example, the page in Figure 18 shows the *linkDown* trap information. For more information on traps see the chapter 'SNMP Primer' on page 76.

Finding out which Rules and Actions Apply to a Specific Trap

To find out which rules and actions apply to a trap selected in the **Traps** list box, click the **Show Rules For The Trap** button on the **SNMP Traps** page. The following page appears:

Rules

Use this page to review the current Trap Console rules applying to a specific trap and/or address.

Display All Rules

Trap: enter the trap name/object ID:

or select below:

MIB:

Enterprise:

Traps:

Origin:

Display Selected Rules

1. [Move up](#), [Move down](#), Move to: [first](#), [last](#)
 Rule: [New rule](#)
 If any trap arrives from SNMP agent at 127.0.0.1, do nothing.

2. [Move up](#), [Move down](#), Move to: [first](#), [last](#)
 Rule: [For all traps from all addresses](#) **disabled**

Figure 21, Rules Applying to a Specific Trap

The **Rules** page shows you the current Trap Console rules applying to the selected/displayed **trap** without any limitation of the trap origin. The trap origin means a host address from where the trap arrives.

To see rules limited by the trap origin, specify the host address in the **Origin** list box. Then click the **Display Selected Rules** button and the **Rules** page displays the current Trap Console rules applying to the selected trap arriving from the selected host.

For the purposes of this subsection, a rule named '**New rule**' was designed which works with trap *LinkUp* arriving from IP address 127.0.0.1 (see Figure 21).

It is possible to **enable**, **disable** or **delete** rules applied to the selected trap. Tick the appropriate check boxes and click the **Enable Checked**, **Disable Checked**, or **Delete Checked** button to perform the wished action.

This page allows you to change the trap and/or address selection too. It is suitable if you want to see rules applying to another trap. Type the trap name in the **Trap** edit line or select the trap in the hierarchic MIB-Enterprise-Traps list set. The host can be selected in the **Origin** list box as described previously. By clicking the **Display Selected Rules** button you get a list of rules applying to the new selected trap (enterprise) and/or address.

Rules are sorted as selected in the **Order** section.

From the **Rules** page you can find out all rules defined in Trap Console too. To see them, click the **Display All Rules** button (see [Figure 23](#) in the section 'Rules').

By clicking the **Create New Rule** button you invoke the **Rule editor** to create a new rule (see the section 'Creating a New Rule' on page 30).

Sending an SNMP Trap

To send a trap to a specific host, select the **Send This Trap** button on the **SNMP Traps** page shown in [Figure 18](#). The following page appears (see [Figure 22](#)):

Send SNMP Trap

Trap: linkDown

Send To:

Agent:

Community:

Local Addr.:

Encoding: SNMP v1 Trap
 SNMP v2 Trap
 SNMP v2 Inform Request

Timeout [s]:

Retries:

Protocol: UDP (default)
 TCP

OID: 1.3.6.1.2.1.11.2

Description: A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the

Variable	Value	Description
ifIndex	<input type="text"/>	[OID: 1.3.6.1.2.1.2.2.1.1] INTEGER. A unique value for each interface. value for each interface must remain constant at least from one re-initialization.

Figure 22, Sending an SNMP Trap

On this page the **Trap** field displays the name of the trap to send.

To send it, follow these steps:

- Enter the IP address or name of the host where the trap is to be sent, into the **Send To** field. Use 127.0.0.1 or localhost when sending the trap directly to Trap Console.

Note: Trap Console sends the trap to the default UDP/TCP trap port 162. Using the URL notation for the Host you can have the trap sent to a different port.

Example: in order to send the trap to the UDP/TCP port 5000, use notation 10.0.0.1:5000.

- Enter the IP address or host name, to be used as the agent address in the trap packet, into the **Agent** field. This value is the address of the host where Trap Console runs.

- Enter the desired SNMP community string for the trap into the **Community** field.
- Specify the **Local Address** only if the computer where Trap Console runs have more than one IP addresses. Leave empty (by default).
- Select the desired SNMP version encoding in the **Encoding** section. You can choose either SNMP v1, SMNP v2 or SNMP v2 Inform Request formats.
- Specify the **timeout** in seconds, and the number of **retries** below.
- Select which **protocol** should be used when sending this SNMP trap. UDP protocol is used by default.
- **OID** (Object Identifier) is the unique identification number of the trap.
- **Description** of the trap can be found in the last section of this table.

Trap values:

This table displays all available variables of the trap (if any) and their short description.

- Enter values for trap variables you wish into their respective **Value** fields.
- Finally, to send the trap, press the **Send Trap** button.

At this moment, the specified trap is sent out to the specified address.

Handling Unresolved Traps

Unresolved traps are traps which were not found in the Trap Console MIB database. For those traps Trap Console cannot decode some trap variables, descriptions or values properly.

Unresolved traps were just logged into the log file `unresolved.log` and discarded in Trap Console version less than 1.3. It caused problems especially when new unexpected trap came into Trap Console. If an appropriate MIB files was not present or compiled in, this trap was discarded. Trap Console was not able to forward it to other co-operating systems.

Starting from version 1.3 such traps are handled as any others. You can define rules and/or actions for unresolved traps. If a rule and/or action has no exception to unresolved trap, it will be performed for all incoming traps properly.

Now Trap Console can forward unresolved traps to other co-operating systems even if appropriate MIB files are not present or compiled in. It means you do not need to have MIBs for all possible incoming traps. In fact you can be never sure about all possible types of incoming traps. If Trap Console has the rule to forward all traps, it forwards them without any losses.

In addition, unresolved traps are logged into the log file `unresolved.log` as in the previous Trap Console versions. You can use this information in order to locate the proper MIB file and add this MIB file to Trap Console's MIB database.

Trap Statistics

Trap statistics for 3rd party monitoring purposes is implemented. The result of monitoring is an XML-encoded document with trap and thread statistics presented on the '/TrapStats.class' page. This page can be accessed without any need to log in, as for example:

```
http://localhost:6610/TrapStats.class
```

provided the client browser runs on `http://localhost:6610`.

To monitor TCP trap receiver and syslog receiver statistics, second XML document is available at:

```
http://localhost:6610/RecvStats.class
```

It contains statistics for all four receivers, structured similarly as the corresponding **Runtime** page.

Rules

Rules in Trap Console can be considered as definitions containing information on **how to manage incoming traps**. In other words, rules contain hints on what to do when a specified trap arrives. Rules are each stored in separate `RULE_*.cfg` files in the `TrapConsole` folder.

Although the rules are editable with Microsoft Windows Write editor, Trap Console Manager offers a more comfortable way to edit and manage rules using the **All Rules** page, see next.

Reviewing Rules

The **All Rules** page (see Figure 23) allows you to review and manage rules used to handle SNMP traps.

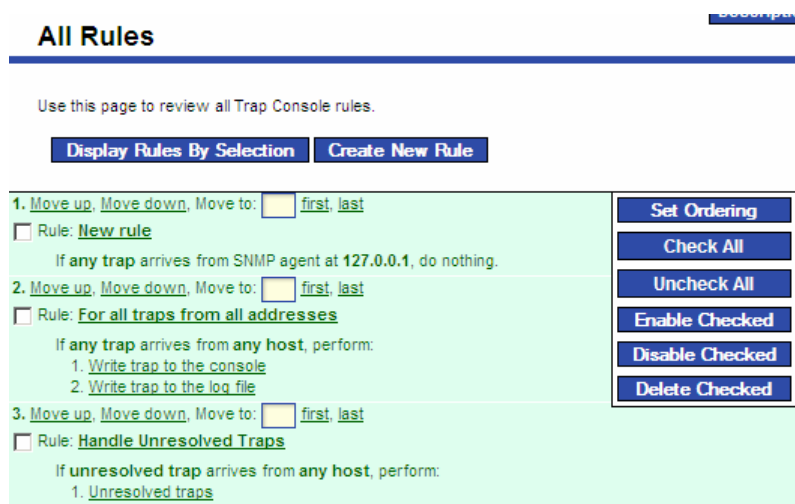


Figure 23, All Rules Page

By default, all rules are displayed. However, pressing the **Display Rules by Selection** button and specifying certain criteria (see Figure 24), only the specified rules will be then displayed.

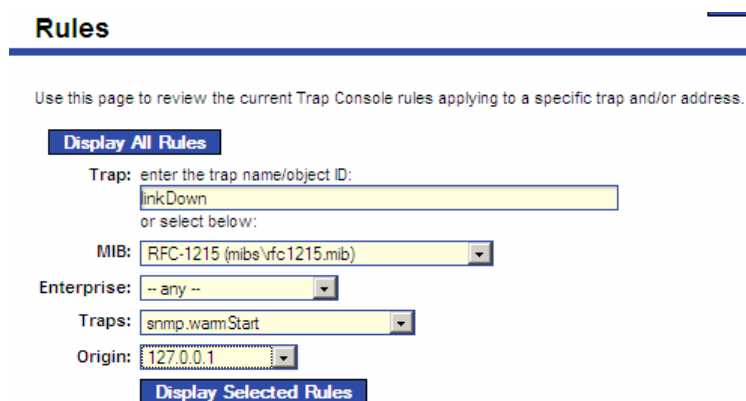


Figure 24, Selecting rules to display

Each of the rules has its own name and description and occupies one line in the bottom frame. The name is a link to a rule editor page for the rule (see Figure 25). Rules can be organized using controls at the top of the rule line (**Move up, Move down, Move to:, first, last**) Rule order is relevant and is used in rule evaluation.

After installing Trap Console two pre-defined rules are automatically created (see Figure 23):

- **For all traps from all addresses** – This rule handles any trap arriving from any host. Trap Console writes all traps to the log file `traps.log` regardless of resolving them.

- **Handle Unresolved Traps** - This rule handles unresolved traps. These traps are written to the log file `unresolved.log` according to the rule. You can use this information in order to locate the proper MIB file and add this MIB file to the Trap Console's MIB database.

Trap Console enables to create arbitrary new rule. Let us look at this in more details.

Creating a New Rule

To create a new rule, invoke the **Rule Editor** by clicking the **Create New Rule** button at the **All Rules** page (see [Figure 23](#)). Creating a new (or editing an existing) rule consists of seven steps:

- **General** - general rule settings
- **Filter** - filtering traps
- **Expressions** - filtering traps conditionally by expressions
- **Addresses** - filtering addresses
- **Actions** - setting actions
- **Calendar** - setting the time and day when the trap is enabled
- **Deduplication** - specifying 'deduplication'

Commit changes by pressing the **Apply** button.

To finish creating the rule at any point, press the **Finish** button. It will save the entered settings and will redirect you to the **All Rules** page.

General Settings

- Change the rule name by typing a new one in the **Name** edit box and pressing the **Rename this Rule** button at the bottom of the table.
- **Enable/Disable** the rule immediately after creating (or editing) by pressing the respective button,
- Specify a short description of the rule in the **Comment** text area.

Rule Editor

General Filter Expression Addresses Actions Calendar Deduplication

Name: New rule

Summary: If any trap arrives from SNMP agent at 127.0.0.1, do nothing.

Name:

Comment:

Rename Enable Copy Delete...

Stop further processing

Apply Cancel Finish

Figure 25, Rule Editor - General

This page also allows you to create a **Copy** of this rule or **Delete** it by clicking the appropriate button.

You can **Allow next rule evaluation** to continue the processing of rules following in the rule evaluation order if this rule applies, or **Stop further processing**.

When the desired information has been entered, click the **Apply** button.

Filtering Traps

Before answering this question, allow us to state that the rule is initially created in the following form:

If **any trap** arrives from **any host**, do **nothing**.

That means that the newly created rule handles all incoming traps from all hosts but does nothing for now. However, the bold faced words are, during the Rule Editing process, replaced according to the actual rule's settings.

On Filter page (see [Figure 26](#)) you can specify traps/enterprises to be processed by this rule.

Enter the trap name or object ID in the edit box, or select the trap from the hierarchical combo boxes). After pressing the **Add Enterprise/Trap** button, the trap will appear as a link on the **Traps** line.

Note: Clicking on the trap's link will **remove** the trap from this rule.

Rule Editor

General	Filter	Expression	Addresses	Actions	Calendar	Deduplication
---------	--------	------------	-----------	---------	----------	---------------

Name: New rule

Summary: If **snmp** or **csCareTrap** arrives from SNMP agent at 127.0.0.1, with exception of **authenticationFailure**, do nothing.

Traps: [snmp](#), [csCareTrap](#)

enter the trap or enterprise name/object ID:

or select below:

MIB:

Enterprise:

Traps:

Except For: [authenticationFailure](#)

Trap: enter the trap or enterprise name/object ID:

or select below:

MIB:

Enterprise:

Traps:

Severity:

Figure 26, Rule Editor - Filter

For the traps description see the section 'SNMP Traps' on page 78.

Exceptions

In addition, exception to the specified traps/enterprises can be set in the **Except For** section.

To exclude a trap/enterprise:

- Enter the trap object ID or choose the trap (or enterprise) from the combo boxes in the **Except For** section, and click the **Add Exception** button. The trap link appears in the row following the 'Except For' label.

To remove trap exclusion:

- Clicking on the link representing the trap, the exception defined for this trap will be removed.

Severity

Trap Console is able to work with trap severities. Severities are proprietary extensions for SNMP traps. They express priorities for trap handling. Trap Console can detect the following severities:

critical, major, minor, informal, warning, intermediate, and unknown (if the appropriate MIB does not contain any information about trap severities). To get more information about severities look at <http://developer.novell.com/research/appnotes/1998/july/03/03.htm>.

Trap Console Mib Compiler allows to compile a MIB including severities if there are some defined in that MIB.

Trap Console rules can be based on trap severities. You can select a trap for which this rule is applied, according to its severity. For example, a rule can be applied only for traps with the critical severity.

To apply a rule only for a selected severity:

- Select the trap severity level for which the rule should be applied in the combo boxes of the **Severity** section.

Example:

If we have the filter set as on [Figure 26](#), selecting 'equal or higher than' and 'minor' will put the rule in force only if a trap from the selected traps/enterprises arrive and at the same time its severity is higher or equals to minor.

Finally, click the **Apply** button.

Filtering Traps Conditionally by Expressions

You can make this rule handle traps conditionally by defining **expressions** on trap parameters and variables. This allows to limit the rule by values of parameters and variables of traps that come in Trap Console. By default, all traps are accepted. You can make the rule apply only to traps with parameters and variables which cause the given **expression** to evaluate as true.



The screenshot shows the 'Rule Editor' window with several tabs: General, Filter, Expression, Addresses, Actions, Calendar, and Deduplication. The 'Expression' tab is selected. The 'Name' field contains 'New rule'. The 'Summary' field contains 'If snmp or csCareTrap arrives from SNMP agent at 127.0.0.1, with exception of authenticationFailure, do nothing.' Below the summary is a large yellow text area for the 'Expression' field, which currently contains the text 'true'. At the bottom of the window, there is a note: 'By default all traps are accepted. If you want this rule to handle traps conditionally based on the result of evaluation of the expression on trap parameters and variables, enter the expression into the Expression edit box above, and click Apply.'

Figure 27, Rule Editor - Expression

Enter the expression into the **Expression** text field and click the **Apply** button.

Detailed information about **expressions** and the way how to define them can be found in the separate description page, click the **Description** link in the top right corner, or see [Expressions](#).

The rule can be **Enabled/Disabled** immediately after creation (or editing) by using the respective button.

Filtering Traps Based on their IP Addresses

Furthermore, Trap Console can process traps arriving from certain IP addresses only. This is can be set on the **Addresses** page (see [Figure 28](#)).

By default, traps from all IP addresses are accepted. To limit the rule to traps coming from specific IP addresses, add those addresses or host names into the **Addresses** list. Enter the address or address range into the text box and press **Add Address/Range**. Repeat until all required addresses appear in the **Addresses** list. Enter address range in the following format: “*address1..address2*”.

Rule Editor

General Filter Expression **Addresses** Actions Calendar Deduplication

Name: New rule

Summary: If **snmp** or **csCareTrap** arrives from SNMP agent at **192.168.1.2** or **10.0.0.1..10.0.0.8**, with exception of **authenticationFailure**, do nothing.

Addresses: **192.168.1.2, 10.0.0.1..10.0.0.8**

192.168.1.5..192.168.1.8 **Add Address / Range**

Enter address range in the format *address1..address2*

Address Is: SNMP Agent
 Trap Sender

Make rule optionally bound to either agent or sender address. One possible scenario for the latter case is when a particular trap arrives from agent through a trap forwarder. Agent and sender addresses are different then.
Note that SNMP v2 does not support agent address. SNMP v2 traps always use the sender address except for the v1 trap translation by a proxy.

Valid for: Given Addresses
 All Other Addresses

Make rule optionally bound to addresses given in the **Addresses** field above, or to all addresses except for the given ones.

Apply **Cancel** **Finish**

Figure 28, Rule Editor - Addresses

If the address is specified, the words **any host** in the rule formula are replaced by that address, e.g.

If any trap arrives from **127.0.0.1**, do nothing

Note: Clicking on the address (range) link will **remove** the address (range) from the list.

In the **Address Is** section, select whether you wish to make the rule bound to either an SNMP Agent or a Trap Sender address.

Note: SNMP v2 does not support agent address. SNMP v2 always use the sender address regardless of the rule settings here.

In the **Valid For** section, select from the displayed options whether to make the rule optionally bound to addresses entered in the **Addresses** text field, or to all addresses except for the entered ones.

Finally, click the **Apply** button.

Assigning an Action to a Rule

Trap Console is capable of operating with **Actions**. Actions are simply descriptions of what Trap Console must do. Actions are stored in separate ACTION_*.cfg files in the TrapConsole folder (where the asterisk (*) is replaced by the Action Name).

On the **Action** page (see [Figure 29](#)) of the Rule Editor you can specify **which Actions** (from the already created ones placed in the TrapConsole folder) should be performed by this rule.

Note: Actions are **created** and **edited** in another part of Trap Console - using the **Action Editor** (see the section 'Actions' on page 38).

Select an action from the **Action Name** combo box to assign to the edited rule. Then select whether it should be enabled/disabled in the **Enabled** checkbox, set the **timeout**, and then click **Add/Update**.

The added action appears as a link under the combo box, together with its status and the set timeout.

[Add/Update Rule](#)

Rule Editor

General	Filter	Expression	Addresses	Actions	Calendar	Deduplication
---------	--------	------------	-----------	---------	----------	---------------

Name: New rule

Summary: If snmp or csCareTrap arrives from SNMP agent at 192.168.1.2 or 10.0.0.1..10.0.0.8, with exception of authenticationFailure, if trap severity equal CRITICAL, perform:

- [Write trap to the console](#)
- [Write trap to the log file \(disabled\)](#)

In case the pre-conditions for this rule were met, perform the following actions

Enabled	Action Name	Timeout	
<input checked="" type="checkbox"/>	Write trap to the console	5 min.	Add/Update
enabled	Write trap to the console	5 minutes	Delete
disabled	Write trap to the log file	0 minutes	Delete

Figure 29, Rule Editor - Adding a New Action to the Rule

In order to prevent 'action storms' when multiple identical traps are received, you can enter an action **timeout**. The timeout guarantees that the specific action performs only once within the specified timeout. The set timeout applies to the edited rule. You can have an action selected for multiple rules with different timeouts.

To **enable** or **disable** any of the listed actions, click on the action name (hyperlink) you wish to modify first, tick or clear the **Enabled** check box, and press the **Add/Update** button on the right. Apply this procedure for each action individually.

Changing timeout for particular actions is analogous.

To **remove** an action from the list, click on its respective **Delete** link.

To apply changes, press the **Apply** button.

Limiting Rules Performing by the Time/Day

On the **Calendar** page (see [Figure 30](#)) you can set the time and day when this rule will be enabled. By default, the rule is performed at all times if enabled.

Description

Rule Editor

General	Filter	Expression	Addresses	Actions	Calendar	Deduplication
---------	--------	------------	-----------	---------	----------	---------------

Name: New rule

Summary: If snmp or csCareTrap arrives from SNMP agent at 192.168.1.2 or 10.0.0.1..10.0.0.8, with exception of authenticationFailure, if trap severity equal CRITICAL, perform:

1. [Write trap to the console](#)
2. [Write trap to the log file \(disabled\)](#)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Monday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tuesday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Wednesday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Thursday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Friday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Saturday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sunday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Check All
Uncheck All

Figure 30, Rule Editor - Calendar

To limit the rule performing by the time/day:

- Select the day and the time when this rule will be enabled by checking/unchecking the appropriate check boxes in the calendar.
- Click the **Apply** button.

Deduplication - How to Prevent Double Processing of Identical Traps

In certain cases, specific devices and applications may generate repeatedly a set of identical and relevant traps. In some cases it is required to react only to the first trap out of this series of traps, and filter the rest of incoming traps out in order to prevent the execution of identical operations.

With 'deduplication' enabled (see [Figure 31](#)), Trap Console reacts, performing the displayed rule if enabled, only to the first received trap, whereas the following trap from the same Agent Address and the same Trap Name is ignored. There is an option to further differentiate traps based on variable values of particular traps with the same variable name.

The ignored incoming traps are stored for further filtering in the 'deduplication list' (per rule, and eventually in multiple lists for each trap where deduplication based on variable values is applied).

Rule Editor

General	Filter	Expression	Addresses	Actions	Calendar	Deduplication
---------	--------	------------	-----------	---------	----------	---------------

Name: New rule

Summary: If snmp or csCareTrap arrives from SNMP agent at 192.168.1.2 or 10.0.0.1..10.0.0.8, with exception of authenticationFailure, if trap severity equal CRITICAL, perform:
 1. [Write trap to the console](#)
 2. [Write trap to the log file \(disabled\)](#)

Deduplication: Filtering of traps identified as duplicates of previous traps is disabled [Enable Deduplication](#)

Auto Clear: [Minutes] Time interval for clearing the collected duplicate trap list.

[Apply](#) [Cancel](#) [Finish](#)

Figure 31, Rule Editor - Deduplication

At the top of the table, the rule **name** is displayed. The next section displays **summary** information about the rule and the set actions. The listed actions can be edited by clicking on their links.

To **enable/disable** the **rule**, press the button under the summary text.

The **Deduplication** section allows to **enable/disable** deduplication by using the respective button.

Auto Clear time specifies the time period in minutes which if expires, automated deletion of ignored traps and the defined action(s) will be performed. The specified time period is related to the 'Time of Last' column for every 'group' of traps displayed in the Trap List including 'variable names' levels if available.

After setting the options, press **Apply**. To go back to the **All Rules** page and save the changes, press **Finish**. Press **Cancel** to leave this page without applying any changes.

Variables:

Identifying traps (or enterprises) as duplicates can be based on the trap name as well as on the variable values and can be specified in more details in the table below:

Variable level deduplication

Traps will be identified as duplicates based not only on the trap name but also on values of the selected variables. Specify a trap and its variables which values will be used to decide whether the trap is a duplicate of some previous trap.

Enterprise: [Select](#)

Trap: [Select](#)

Variable: [Add](#)

csCareTrap: [alarmCategory](#), [nodeName](#) [Select](#)

linkDown: [ifindex](#) [Select](#)

[Apply](#) [Cancel](#) [Finish](#)

Figure 32, Variables

- Choose the trap you wish to deduplicate from the selected **Enterprise** and **Trap** combo box and press the **Select** button (see Figure 32).
 - Traps (or trap enterprise) for deduplication can be further specified by the **Variable Name**. The respective combo box lists all the possible variable names of the selected trap. Choose one from the combo box and press **Add**. The selected variable will be then listed under the combo box. Traps can have more variable names defined for deduplication.
- Note:** Clicking on a trap **variable name's link**, the variable name will be **removed** from the list!
- Use the **Select** link to pre-fill the trap name in the combo boxes.

To save all performed changes, press **Apply**. To go back to the **All Rules** page and save the changes, press **Finish**. Press **Cancel** to leave this page without applying any changes.

Trap List:

The list of recorded traps can be viewed under the **Trap List** table by pressing the **Show Trap List**. The **Hide Trap List** will hide this list.

- **Number of Traps** is the total number of recorded traps.
- Press **Refresh** to update the latest information on the page.
- Pressing **Clear All** clears the entire list of received traps.

After the Trap list is shown on the page (see [Figure 33](#)), the following information can be found there.

Trap List

Number of Traps: 2 [Refresh](#) [Clear All](#) [Hide Trap List](#)

[Check All](#) [Uncheck All](#) [Clear Checked](#)

Agent Address	Trap Name	Count	Time of First	Time of Last
<input type="checkbox"/>	csCareTrap	0		
<input type="checkbox"/>	linkDown	2		
Agent Address	Trap Name	Count	Time of First	Time of Last

Figure 33, Trap List

- The set **agent address**, **trap name**, their **count**, and the **time** of the **first** and the **last** arrived trap.
- The format of the date and time displayed in columns 'Time of Last' and 'Time of First' is: YYYY:MM:DD HH:MM:SS
- To delete any of the traps in the list, check its respective check box, and press **Clear Checked**. It is possible to **check/uncheck** all traps in the list using the appropriate links.
- Traps can be displayed on more pages provided the number of traps in the list exceeds one page. Use the navigation buttons to navigate between the pages.

If a particular trap has more variable names selected for deduplication, click on the **trap name's link** to view the list of received traps based on the set variable names (see [Figure 34](#)).

linkDown: Variable Level Deduplication Trap List

Agent Address	Trap Name	ifIndex	Count	Time of First	Time of Last
<input type="checkbox"/> 192.168.1.1	snmp.linkDown	2	1	2005.07.19 14:34:36	2005.07.19 14:34:36
<input type="checkbox"/> 192.168.1.1	snmp.linkDown	4	1	2005.07.19 14:34:51	2005.07.19 14:34:51
Agent Address	Trap Name	ifIndex	Count	Time of First	Time of Last

[Check All](#) [Uncheck All](#) [Clear Checked](#)

Figure 34, The Selected Trap

In this list, the trap variable name value is displayed in its respective column in addition to the information displayed in the 'Trap List'.

Example of deduplication:

Deduplication is aimed to be used to eliminate multiple information being further processed:

Let's say you have a device sending LinkDown traps every minute when in failure. You need to know the state of the device all the time if in failure, whereas the service crew, for example, needs to receive the failure information only once to start solving the problem.

So, you have defined a **rule** with an action in Trap Console which sends the 'failure' information via e-mail to you. As you prefer to receive this information at least every ten minutes, you have a ten-minute timeout set in your action.

For the service crew, on the other hand, a **rule** with an action where **deduplication** is enabled should be defined. In Deduplication, Auto Clear should be set for the time during which they do not need to receive this information as the problem is being solved, e.g. 20 minutes.

When the first LinkDown trap arrives, it triggers the set action and the service crew receives the 'failure' message. Deduplication will prevent actions to respond to the consecutive incoming traps. If no LinkDown trap arrives within 20 minute after the last LinkDown trap has arrived, the Auto Clear function will clear the collected duplicated traps. The next received LinkDown trap will trigger the set action again and the 'failure' message will be sent.

Note: Please do not set timeout and deduplication which may collide in one rule.

Actions

In Trap Console, actions can be set to be performed upon reception of a trap.

Actions are represented by separate files containing special 'hints' to Trap Console stored in ACTION_*.cfg files and located in the *TrapConsole* folder.

An example of an Action named 'Write trap to the console' is shown below:

```
#Action Write trap to the console
#Fri Aug 15 11:32:16 GMT+00:00 2006
line1=%time%: %trapName% from
%agentName%(%agentAddress%): %varvals%
className=TrapActionConsole
name=Write\ trap\ to\ the\ console
```

While working with actions, there is no need to access these files. Actions are fully manageable from the web browser interface.

Macros

Action's body is composed of **macros** (delimited with % characters). Macros give you access to some important SNMP variables and Trap Console data. To create new actions you should be familiar with them. To view the detailed description of all available macros, see [Macros](#)

on page 86.

Reviewing an Action

The **Actions** page (see [Figure 35](#)) allows you to review and manage actions established in Trap Console. This page appears when you click the **Actions** hyperlink on the Trap Console Navigation Bar.

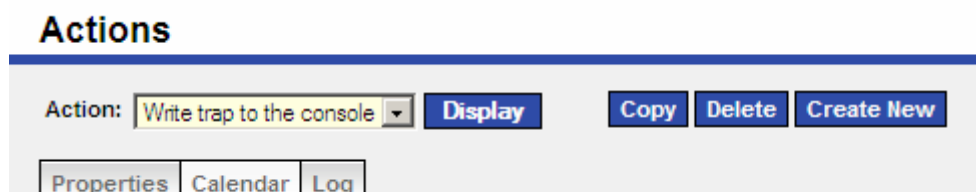


Figure 35, Actions Page

In the **Action** list box, all Actions present in the TrapConsole folder are listed. On this page, actions can be **displayed, copied, deleted** and **created**.

To create a copy of an existing action, click the **Copy Action** button. Then the copy of the original action is created named after the copied action but with 'Copy of' prefix. To apply changes, click the **Update Action** button at the bottom of the page.

To delete the selected action, click the **Delete Action** button.

In order to limit or set the time and day when an action will be enabled, select the **Calendar** tab. A 'time table' with checkboxes appears (see [Figure 36](#))

Description

Actions

Action: Write trap to the console
Display
Copy
Delete
Create New

Properties
Calendar
Log

Calendar

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Monday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Tuesday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Wednesday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Thursday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Friday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Saturday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sunday	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Check All
Uncheck All
Set

Figure 36, Calendar

Select the day and the particular times when the action shall be enabled. By default all days and times are selected. To **check all/uncheck all** times, use the respective buttons. To apply changes, press **Set**.

Each action can log its operation in a separate log file set. Select the **Log** tab (see [Figure 37](#)) This logging is initially disabled. Logging of the selected action can be **enabled/disabled** by pressing the respective button. The log file **size limit** is set to 100 kB by default, but can be changed. To apply changes, press the **Set Log Limit** button. To delete the log file, press the **Delete Log** button. Log content is dumped in a frame below the form.

Actions

Action:

[Log](#)

Logging enabled:

Log File Size Limit: [kB]

```

July 19, 2005 3:19:56 PM CEST: Performing action for:
Trap Name:      linkDown
Agent Address:  192.168.1.1
Sender Address: 127.0.0.1
SNMP Version:  SNMPv1
Time Stamp:    17:6:21.20
Community:     trap
Enterprise:    snmp
Enterprise ID: 1.3.6.1.4.1.5979.1
Severity:      unknown
Variables:
  ifIndex: 4
  
```

Figure 37, Action Log

After installing Trap Console, three pre-defined actions are automatically created:

- **Unresolved Traps** - writes the selected trap information to the log file `unresolved.log`.
- **Write trap to the console** - writes the selected trap information to the Trap Console application window.
- **Write trap to the log file** - writes the selected trap information to the log file `traps.log`.

These pre-defined actions are used in the pre-defined rules. You can create your own rules with these actions or create a new action for a new rule. Let us look at this in more details.

Creating a New Action

By clicking the **Create New Action** button on the **Actions** page shown in [Figure 35](#), the **Create New Action** page appears (see [Figure 38](#)):

Create New Action

Enter the name of the new action, choose its type, then choose the **Create** button.

Choose the **Back** button of your web browser to close this page without creating a new action.

Name:

Type:

Figure 38, Creating a New Action

Here, simply type the Action Name and select the Action Type from the following **Type** list:

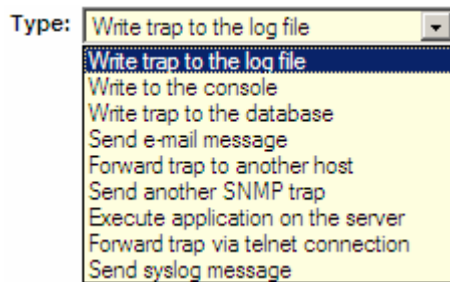


Figure 39, Action Types List Box

Note: The Action Name can contain up to 255 characters, including spaces. It cannot use the following characters: \ / : * ? " < > |.

As you can see, Trap Console is well equipped with various types of actions. Their description can currently serve as an answer to some frequently asked 'how to' questions:

Writing Traps to the Log File

To create a new action which writes trap information to the log file, type the name in the **Name** edit box, select the **'Write trap to the log file'** item from the **Action Types** list box shown in [Figure 39](#) and click the **Create** button.

The following page appears (see [Figure 40](#)):

Figure 40, Writing the Selected Trap to the Log File

On this page:

- **Action Name** can be changed.
- The log file's **File name** can be changed. The file name can contain macros, and thus eventually create a log file hierarchy. Macros expanding in the file name can be used to distribute logs in subdirectories according to trap attributes, such as enterprise, sender, time and date.

Example: %agentName%/traps.log

- **Size limit** of the file where logs will be written can be specified.

- In the **Date Time** section's checkbox, select whether to insert the current data and time automatically before particular entries are written in the log file.
- The **Lines to print to the log file** text area contains macros that define the required information to be written to the log file.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

The log file can be viewed on the **Logs** page (see [Figure 52](#)).

Writing Traps to the Console Window

To create a new action which writes trap information to the Trap Console window, type the name in the **Name** edit box, select the **'Write to the console'** item from the **Action Types** list box in [Figure 39](#) and click the **Create** button.

To view the already created action, press the **Display** button.

The following page appears (see [Figure 41](#)):

The screenshot shows the 'Actions' management interface. At the top, there's a header 'Actions' with a 'Display' button and a dropdown menu showing 'Write trap to the console'. Below this are buttons for 'Copy', 'Delete', and 'Create New'. A secondary bar contains 'Properties', 'Calendar', and 'Log' buttons. The main content area is titled 'Properties' and contains the text: 'Upon reception of an SNMP trap, this action prints the contents of the edit box to the Trap Console window.' Below this, the 'Action Name' is 'Write trap to the console'. The 'Lines to print to the console' text area contains the macro: '%time%: %trapName% from %agentName%(%agentAddress%): %varvals%'. At the bottom, there are 'Update Action' and 'Test Action' buttons.

Figure 41, Writing to the Trap console Application Window

On this page:

- **Action name** can be changed.
- Text to be printed in the console can be changed in the **Lines ton print to the console** text area.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Writing Traps to the Database

To create a new action which writes trap information to any JDBC compliant database, type the name in the **Name** edit box, select the **'Write trap to the database'** item from the **Action Types** list box in [Figure 39](#) and click the **Create** button.

The following page appears (see [Figure 42](#)):

Actions

Action:

Properties

Upon reception of an SNMP trap, this action executes the specified SQL command in the selected database.

Action Name:

JDBC Driver:

Database URL:

User:

Password:

SQL Statement:

Conn. Min.: [sec]

Conn. Max.: [sec]

Timeout: [sec]

Figure 42, Writing to the Database

On this page:

- **Action name** can be changed.
- Check the name of the JDBC driver in the **JDBC Driver** edit box.
*Trap Console is able to detect the type of the Java virtual machine it is running in. Depending on the used JVM, Trap Console suggests in the **JDBC Driver** edit box:*

```
sun.jdbc.odbc.JdbcOdbcDriver
- if you use JDK/JRE from Sun Microsystems
```

This driver is suitable for any ODBC database on Microsoft Windows platform. If you do not use ODBC database, enter the name of the correct driver.

- Enter the database URL into the **Database URL** edit box. Filling this box is required.

A database URL provides the way of identifying a database. The appropriate driver will recognize the database and establish a connection with it. The standard syntax for a database URL has three parts, which are separated by colons:

```
jdbc:<subprotocol>:<subname>, where
```

jdbc is the protocol used to access information. The protocol in the JDBC database URL is always jdbc.

<subprotocol> is the name of the driver or the name of the database connectivity mechanism. For example subprotocol "odbc" is used for an access to ODBC databases (on Microsoft Windows platform).

<subname> is the way to locate the database.

- Enter the database user name into the **User** edit box. (Optional item).
The database user name is your user name for the access to the database.

- Enter the database user's password into the **Password** edit box. (Optional item).
The user password is your password for the access to the database.
- Enter the SQL statement to be performed into the **SQL Statement** edit box. This item is required.
Trap Console uses the SQL statement to insert a new record to the database. SQL statement syntax differs according to the used database. You need to know the correct syntax of the inserted statement. Later we show some examples of SQL statements.
- Enter the minimum number of open database connections into the **Conn. Min.** edit box (Optional item).
Trap Console will maintain a specified number of open connections with the database. This can be for performance reasons or any other JDBC implementation specifics.
- Enter the maximum number of open database connections into the **Conn. Max.** edit box (Optional item).
Trap Console will limit the number of open connections with the database to the specified number. Choose the limit consistent with JDBC or database settings.
- Enter the database connection inactivity timeout in seconds into the **Timeout** edit box (Optional item).
Database connection will be kept open the specified number of seconds after the last statement execution, to save the connection creation overhead for a next access.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Later in this user's guide, in the section 'Working with Databases' on page 64, you'll see in details how to write traps to several databases.

Creating a New E-mail Action

To create a new e-mail action which sends an e-mail to the specified address, enter the name of the action into the **Name** edit box, select the '**Send e-mail message**' item from the list box in [Figure 39](#) and click the **Create** button.

The following page appears (see [Figure 43](#)):

The screenshot shows the 'Actions' configuration interface. At the top, there is a header 'Actions' and a sub-header 'Properties'. Below this, there is a form for configuring an action. The 'Action' dropdown is set to 'Warning e-mail'. There are buttons for 'Display', 'Copy', 'Delete', and 'Create New'. Below these are tabs for 'Properties', 'Calendar', and 'Log'. The main configuration area is titled 'Properties' and contains the following fields:

- Action Name:** Warning e-mail
- From:** trapconsole@localhost
- To:** admin@company.com
- Cc:** (empty)
- Subject:** Trap %enterprise%/%trapName% from %agentAddress%/%agentNa
- Body:** Trap Console received trap %enterprise%/%trapName%.
Time: %time%
Values: %varvals%

At the bottom of the form are two buttons: 'Update Action' and 'Test Action'.

Figure 43, Sending E-mail Message

On this page:

- **Action name** can be changed.
- Enter the e-mail address of the sender of this e-mail message into the **From** edit box. This item is required.
- Enter the e-mail address of the recipient of this e-mail message into the **To** edit box. Filling this box is required.
- Optionally, enter comma separated e-mail addresses of additional recipients into the **Cc** edit box.
- Enter the subject of the e-mail message into the Subject edit box.
- Enter the message body into the **Body** text area.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Forwarding a Trap to Another Host

To create a new action which forwards a trap to another host, type the name in the **Name** edit box, select the '**Forward trap to another host**' item from the **Action Types** list box in [Figure 39](#) and click the **Create** button.

The following page appears (see [Figure 44](#)):

The screenshot shows the 'Actions' configuration page. At the top, there is a header 'Actions' and a sub-header 'Properties'. Below the header, there is a dropdown menu for 'Action:' set to 'Re-send trap to 192.168.1.2', and buttons for 'Display', 'Copy', 'Delete', and 'Create New'. Below this, there are tabs for 'Properties', 'Calendar', and 'Log'. The main content area is titled 'Properties' and contains the following fields and options:

- Action Name:** Re-send trap to 192.168.1.2
- Host:** 192.168.1.3
- Agent Addr.:** localhost
- Local Addr.:** (empty)
- Encoding:**
 - do not change
 - SNMP v1 Trap
 - SNMP v2 Trap
 - SNMP v2 Inform Request
- Timeout [s]:** 1
- Retries:** 0
- Protocol:**
 - do not change
 - UDP (default)
 - TCP

Figure 44, Forwarding a Trap to a Different Host

On this page:

- **Action name** can be changed.
- Enter the name or IP address of the host, where the trap will be forwarded, into the **Host** edit box.

Note: Trap Console sends the trap to the default UDP trap port 162. Using the URL notation for the Host you can have the trap sent to a different port.

Example: In order to send the trap to the UDP port 5000, use notation `target:5000`.
- Enter the agent address/name in the **Agent Addr** edit box. Leave the field blank to keep the original agent address during trap forwarding.

- Enter the preferred local interface address into the **Local Addr** field. Leave the field empty if you do not have multi-homed host or do not care which interface is going to be used.
- Select the outgoing trap SNMP version encoding used in the **Encoding** section. The first option means that the used encoding will be the same as of the received trap. Other options will cause translation of trap PDU (Protocol data unit) to their respective encoding.
- **Timeout** field is used with Inform Request encoding type. Its value is the amount in seconds Trap Console will wait for response before attempting retry or reporting timeout error.
- **Retries** field is used with Inform Request encoding type. Its value is the maximum number of repeated requests in case that the response is not received within the set timeout interval.
- Select the outgoing trap transport protocol used in the **Protocol** section. The first option means that the used protocol will be the same as that used with the received trap. Other options will force their respective protocol.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Sending Another SNMP Trap

To create a new action which creates and sends a new SNMP trap, type the name in the **Name** edit box, select the '**Send another SNMP trap**' item from the **Action Types** list box in [Figure 39](#) and click the **Create** button.

The following page appears (see [Figure 45](#)):

Actions

Action: Send linkDown Display Copy Delete Create New

Properties Calendar Log

Properties

Upon reception of an SNMP trap, this action creates a new SNMP trap and sends it to the specified host.

Action Name:

Host:

Agent:

Community:

Local Addr.:

Encoding: SNMP v1 Trap
 SNMP v2 Trap
 SNMP v2 Inform Request

Timeout [s]:

Retries:

Protocol: UDP (default)
 TCP

Trap: enter the trap name/object ID:

 or select below:

MIB:

Enterprise:

Traps: Select

Trap info

Selected Trap: linkDown
 OID: snmp.2 [1.3.6.1.2.1.11.2]

Description: A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.

Trap values

Variable	Value	Description
ifIndex	<input type="text" value="2"/>	[OID: 1.3.6.1.2.1.2.2.1.1] INTEGER. A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

Figure 45, Sending a New SNMP Trap

This action creates a new SNMP trap according to the settings and sends the trap to the specified host.

On this page:

- **Action name** can be changed.
- Enter the name or IP address of the host, where the trap will be sent, into the **Host** edit box.
Note: Trap Console sends the trap to the default UDP trap port 162. Using the URL notation for the Host, you can have the trap sent to a different port.
Example: in order to send the trap to the UDP port 5000, use notation `target:5000`.
- Enter the name or IP address of the host, from which the trap is being sent, into the **Agent** edit box. This should be the domain name or IP address of the local host.

- Enter the trap community string into the **Community** edit box. Some hosts may reject traps with incorrect community.
- Select the desired SNMP version encoding in the **Encoding** section. You can choose the SNMP v1, SNMP v2 or SNMP v2 Inform Request format.
- **Timeout** field is used with Inform Request encoding type. Its value is the amount in seconds Trap Console will wait for response before attempting retry or reporting timeout error.
- **Retries** field is used with Inform Request encoding type. Its value is the maximum number of repeated requests in case that the response is not received within the set timeout interval.
- Select the outgoing trap transport protocol used in the **Protocol** section. The first option means that the used protocol will be the same as that used with the received trap. Other options will force their respective protocol.
- Choose the trap to send from the **Trap** list box and press **Select**.
- The **Description** field shows the description of the selected trap.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Executing an Application on the Server

To execute an application on the server where Trap Console is running, type the name in the **Name** edit box, select the '**Execute application on the server**' item from the **Action Types** list box in Figure 39 and click the **Create** button. The following page appears (see Figure 46):

Actions

Action: Run backup

Properties

This action executes the selected external application as a result of the received SNMP trap. Note that the action will be executed on the server where Trap Console is running!

Action Name:

Path:

Parameters:

Environment variables

Following environment variables will be added to an external application's environment

Variable Name	Variable Value	
<input type="text"/>	<input type="text"/>	<input type="button" value="Add/Update"/>
SYSTEM_DRIVE	C:	<input type="button" value="Delete"/>

Figure 46, Executing an External Application on the Server

On this page:

- **Action name** can be changed.
- Enter the path to the executable file into the **Path** edit box.
- Enter the executable parameters into the **Parameters** edit box.
- Set the external application's environment in the **Environment variables** section by specifying the **Variable Name** and **Variable Value** in the respective edit boxes. After pressing **Add/Update**, the set variable appears under the edit boxes. To delete the variable, click its respective **Delete** link.

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Forwarding a Trap via Telnet Connection

To create a custom socket communication script which forwards information about a trap to a different host via TCP connection, type the name in the **Name** edit box, select the '**Forward trap via telnet connection**' item from the **Action Types** list box in Figure 39 and click the **Create** button.

The following page appears (see Figure 47):

The screenshot shows the 'Actions' management interface. At the top, there is a dropdown menu for 'Action:' set to 'Telnet connection', with buttons for 'Display', 'Copy', 'Delete', and 'Create New'. Below this are tabs for 'Properties', 'Calendar', and 'Log'. The 'Properties' tab is active, showing a description: 'Upon reception of an SNMP trap, this action establishes a TCP (socket) connection with the specified host and socket. After connecting, it performs the script and disconnects.' The configuration fields are: 'Action Name:' (Telnet connection), 'Host:' (192.168.1.3), 'Socket:' (empty), and 'Script:' (Time: %time%, trap %enterprise%/%trapName%, from %agentAddress%/%agentN...). At the bottom are 'Update Action' and 'Test Action' buttons.

Figure 47, Forwarding a Trap via TCP Connection

On this page:

- **Action name** can be changed.
- Enter the host name or IP address, where the connection will be established, into the **Host** edit box.
- Into the **Socket** edit box, enter the TCP socket (number) of the server process on the host to which this action will connect.
- The **Script** describes the communication to carry out with the server. The whole communication is line oriented.

To send a line to the server, start it with the keyword `send:` (e.g. `send:hello`).

To check for the correct reply from the server, enter a line with the expected answer. This line must start with the keyword `recv:.` The action will test that the response from the server starts with the string that directly follows the `recv:keyword`.

Example: here is a script that performs an anonymous login into an ftp server:

1. `recv:220`
2. `send:user anonymous`
3. `recv:331`
4. `send:pass trapgate@cscare.com`
5. `recv:230`

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Logging trap information to a remote syslog server

To send syslog message, type the name in the **Name** edit box, select the '**Send syslog message**' item from the **Action Types** list box in Figure 39 and click the **Create** button. The following page appears (see Figure 48):

Actions

Action: Syslog **Display** **Copy** **Delete** **Create New**

Properties **Calendar** **Log**

Properties

Upon reception of an SNMP trap, this action uses trap information to build syslog message and forwards it to a remote Syslogd server.

Action Name: Syslog

Facility: random user-level messages Use received syslog message facility (csSyslogTrap variable "facility")

Severity: Informational: informational messages Use received syslog message severity (csSyslogTrap variable "severity")
 Select severity automatically according to --#SEVERITY clause in the MIB

Timestamp: %timeticks%

Originating Host: %agentAddress%

Tag: trapconsole

Content: trap %enterprise%/%trapName%, from %agentAddress%/%agentName%

Protocol: UDP (default) TCP

Host: localhost

Port: 514

Local Address: -- any --

Update Action **Test Action**

Figure 48, Sending syslog message

On this page:

- **Action name** can be changed.
- Select syslog message priority by combination of **Facility** and **Severity** selections. If the action is going to be used in a rule matching **csSyslogTrap** (trap generated upon reception of syslog message by Trap Console) there are two checkboxes available to reuse its facility and severity variables ('**Use received syslog message facility**' and '**Use received syslog message severity**' checkboxes)
- **Timestamp** edit box by default contains timeticks macro and is substituted by current timeticks value at runtime. There is little reason to use something else, constant value would cause syslog messages with identical time stamp.
- **Originating Host** edit box value will be used as a syslog message source identification. You can use the default %agentAddress% macro for trap address or something constant like %gateAddress%.
- **Tag** value is a string used to differentiate messages coming from the same originating host, for example when facility and severity do not suffice.

- **Content** is the syslog message content. Basic trap information is packed there by default.
- **Protocol** options are UDP and TCP, UDP means using an UDP packet for a single message, the default syslog transport type. With TCP option, a TCP connection is opened for reliable message transport. An open connection will close automatically when idle for a long time. It is still a BSD syslog protocol in both cases.
- **Host** and **Port** define the remote syslog server socket where the generated messages will be forwarded.
- If you have any preferable **Local Address** to use for sending syslog messages out, select it from the list.

Performing an SNMP Set request

To send SET request, type the name in the **Name** edit box, select the '**Perform SNMP Set request**' item from the **Action Types** list box in [Figure 39](#) and click the **Create** button. The following page appears (see):

The screenshot shows the 'Actions' configuration page in Trap Console. At the top, there is a dropdown menu for 'Action' set to 'Set request', with buttons for 'Display', 'Copy', 'Delete', and 'Create New'. Below this are tabs for 'Properties', 'Calendar', and 'Log'. The main configuration area includes:

- Action Name:** Set request
- Host:** agent2
- Protocol:** UDP (default), TCP
- Local Address:** -- any --
- Timeout [s]:** 1
- Retries:** 0
- Version:** SNMPv1, SNMPv2
- Community:** public
- MIB:** IF-MIB (mibs\IF-MIB.my)
- Variable:** 1.3.6.1.2.1.2.2.1.14 (ifInErrors) Counter32 (INTEGER(0..4294967295))

Below the variable list, there is a table with columns 'Variable', 'Value', and 'Description':

Variable	Value	Description
1.3.6.1.2.1.2.2.1.14.0	%msgNumber%	<input type="button" value="Add"/> <input type="button" value="Remove"/>

At the bottom, there is a 'Variables' section with the entry: 1.3.6.1.2.1.2.2.1.6.0 %senderAddress% and buttons for 'Update Action' and 'Test Action'.

Figure 49, Performing Set request

On this page:

- **Action name** can be changed.
- Enter the name or IP address of the host, where the trap will be sent, into the **Host** edit box.

Note: Trap Console will send the request to the default snmp port 161. Using the URL notation for the Host, you can have the message sent to a different port.

Example: in order to send the trap to the UDP port 5000, use notation `target:5000`.
- Select the outgoing message transport protocol used in the **Protocol** section.
- Enter preferred local interface address into the **Local Address** box. Leave the field empty if you do not have a multihomed host or do not care which interface is going to be used.
- **Timeout** field value is the amount in seconds a Trap Console will wait for response before attempting retry or reporting timeout error.

- **Retries** field value is the maximum number of repeated requests in case a response is not received within timeout interval.
- Select the desired Set-Request PDU message SNMP version in the **Version** section.
- Enter the request write community string into the **Community** edit box. Some hosts may reject messages with incorrect community.
- Prepare the variable bindings for the Set-Request PDU by filling the **Variable** and **Value** edit lines acknowledged with the **Add** button. Variable can also be selected using drop down lists **MIB** and **Variable** that present OIDs available for individual MIBs. You can modify or remove the variable from the list of PDU variables. To modify the list, select given variable by clicking on its name (hyperlink), edit its definition in the **Variable** and **Value** edit lines and click the **Add** button. To remove the variable from the list, click **Remove** button

To apply changes, press the **Update Action** button, and to test the action press **Test Action**.

Testing an Action

This section describes how to verify an action. In this way you can discover and fix potential problems during the action design stage. We recommend you to test each new action before you use it in real conditions.

To test an arbitrary existing action, select the action in the **Action** list box on the **Actions** page shown in [Figure 35](#). The action's description gets displayed in the lower part of the page. Click the **Test Action** button at the bottom of the action's description section. The following page appears (see [Figure 50](#)):

Select Trap for Action Test

Select a trap which will be used for testing the action. This page displays information about all SNMP traps current MIB files and by trap enterprises.

MIB:

Enterprise:

Traps:

-- any --
linkUp
coldStart
linkDown
authenticationFailure
egoNeighborLoss
warmStart

Trap: linkDown
Enterprise: snmp.2
OID: 1.3.6.1.2.1.11.2
Description: A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.
Severity: unknown
ifIndex [OID: 1.3.6.1.2.1.2.2.1.1] INTEGER. A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

Figure 50, Selecting a Trap for an Action Test

On this page you can select a trap which will be used for testing the previously selected action. Select the desired trap in the **MIB**, **Enterprise** and **Traps** lists. The **MIB** and **Enterprise** lists serve as a filter to display only traps belonging to specific MIB or enterprise. To reload the page with the detailed information about the desired trap, choose the **Show Trap** button. Then click the **Use This Trap** button to use this trap for testing the action. The page allowing you to enter parameters for the selected trap appears (see [Figure 51](#)):

Set Trap Parameters

The action will be tested on the selected trap with parameters entered here.

Trap: linkDown

Agent IP:

Community:

Encoding: SNMP v1 Trap
 SNMP v2 Trap
 SNMP v2 Inform Request

Protocol: UDP (default)
 TCP

OID: 1.3.6.1.2.1.11.2

Description: A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.

[Trap values](#)

Variable	Value	Description
ifIndex	<input type="text"/>	[OID: 1.3.6.1.2.1.2.2.1.1] INTEGER. A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re- initialization.

Figure 51, Setting Trap Parameters

On this page:

- The **Trap** field displays the name of the trap to send.
- Enter the IP address to be used as the trap's agent address into the **Agent IP** field. This value will be set as the default address of the host where Trap Console runs.
- Enter the SNMP community string for the trap into the **Community** field.
- Select the desired SNMP version encoding in the **Encoding** section. You can choose the SNMP v1, SNMP v2 or SNMP v2 Inform Request format.
- Enter values for all trap variables into respective **Value** fields, if the trap has any variable values.
- Click the **Test Action** button to start testing the action.

If your action has been defined properly, the following message should appear:

Test Action - Result

The action was tested with the following result:

OK

If not, then return to the **Actions** page (see [Figure 35](#)) and look at macros defining the tested action. Try to write them correctly or look for other possible problems such as access rights to databases, busy UDP port, TCP socket etc. Potential problems depend on the action definition.

To apply changes, click the **Update Action** button.

Note: Each time you make changes on the **Actions** page, it is necessary to apply them by the **Update Action** button. Without it, new changes have no influence on the selected action.

To start the test again, press the **Test Action** button.

Distributing New Rules and Actions to Multiple Servers Easily

To distribute existing rules and actions to multiple servers, just copy the `RULE_*.cfg` and `ACTION_*.cfg` files to a different Trap Console installation and restart Trap Console there.

Log Files

Logs generated by actions of type **'Write trap info to the log file'** are accessible on this page. ([Figure 52](#)). **Action** combobox lists all such actions. **Log File** combo box lists log files created by the selected action or all log files if no action is selected (in this case the log file name is appended with action name in parentheses)



Figure 52. Logs Page

After installing Trap Console, several actions writing to log files are automatically created as well as their corresponding log files:

- `traps.log` – it holds a list of all incoming traps.
The action **'Write trap to the log file'** writes incoming traps to this file.
- `unresolved.log` – it holds a list of all incoming traps which are not resolved by the Trap Console MIBs database. This log file is created only after the first unresolved trap is received.
The action **'Unresolved traps'** uses this file.

To view a particular log file, select the action to which the log file belongs from the **Action** combo box and press **Select**. Then the **Log File** combo box contains only log files which belong to the selected action.

Select the required log file from the **Log File** combo box and press **Display Log**. The content of the selected log file will be displayed in the bottom frame. Press **Clear Log** to clear the **contents** of the selected log file.

Press **Delete Log** to completely **remove** the selected log file.

Autorefresh of the selected log file is enabled if the **Enable** checkbox is ticked. The autorefresh rate can be specified in the respective edit box. To apply the settings, press **Set**.

Note: A new handling of log files was introduced to prevent Trap Console log files from growing forever. Log file sizes are limited to 100 KB except the `smtpsender.log` file. Its size is limited to 1000 KB. You can set a size according to your need in the `trapconsole.cfg` file (see [Customizing trapconsole.cfg](#)). Log files internally maintain the limited size by overwriting the oldest records. They are divided into smaller parts, marked as `*.LOG.x`, where `x` means the `x`-th part of the log file. There can be at most ten parts of one log file at the same time.

User Management

Trap Console allows you to assign individual users specific access rights. The idea is that only the administrator and selected users can modify Trap Console rules and/or actions. Users with read-only access rights assigned are allowed to view the created rules/actions and log files only.

Trap Console comes with two built-in accounts: the administrator and the guest.

The 'administrator' has complete access rights to proceed with all Trap Console operations. After installation, the administrator account is used as the default user for Trap Console. This account has an empty password at this time. It enables you to get in easily with complete access rights. Later you or your Trap Console administrator can learn how to create other users, change passwords and limit access rights.

It is highly desirable that the Trap Console administrator changes the password as soon as possible.

The second pre-set user account 'guest' has also an empty password after installation. Guests cannot change this password. In addition, guests cannot create, modify or delete rules and/or actions, use the MIB compiler or change Trap Console settings.

Another two user account types can be added to Trap Console:

- the read-write user - he/she can create, modify and delete rules and actions, compile MIBs, tune Trap Console performance and change his/her password. In fact, this user can do all the same operations as the administrator except for creating/deleting user accounts.
- the read-only user - he/she can view the previously created rules/actions and monitor log files to see what has happened. This account type allows to change his/her password.

Note that it is not possible to have multiple users logged in at the same time under the same login name (except for the guest).

This page enables you to create new users with appropriate access rights, edit them or delete some of them (see [Figure 53](#)). Only the Trap Console administrator can change the user management system.

The screenshot shows the 'User Manager' interface. At the top, there is a 'Preferences' section with tabs for 'Licensing', 'Application Log', 'Address Restrictions', 'Restricted Trap Log', and 'Mapping'. Below this, the 'User Manager' title is centered. A subtitle reads: 'Use this page to create or delete users and set the user access rights.' The main area contains a table with columns for 'User Name', 'Read-only Access', and 'Actions'. There is an input field for a new user name, a checked checkbox for 'Read-only Access', and an 'Add / Update' button. The table lists three users: 'administrator' with 'write' access, 'quest' with 'read only' access, and 'user1' with 'read only' access. Each 'read only' user has 'Edit' and 'Delete' buttons.

User Name	Read-only Access	Actions
<input type="text"/>	<input checked="" type="checkbox"/>	Add / Update
administrator	write	
quest	read only	Edit Delete
user1	read only	Edit Delete

Figure 53, User Manager

To create a new user account:

- Type the name for a new user in the **User name** edit line.
- Tick the **Read-only Access** check box to set the user access rights to read-only.
- Click the **Add/Update** button.

The new user name appears in the **User Name** list including his/her access rights.

To edit user access rights:

- Click the **Edit** hyperlink of the user you wish to edit or type the user name in the **User Name** edit box.
- Change the access rights according to your requirements.
- Click the **Add/Update** button.

To delete the user:

- Click on the **Delete** hyperlink of the user you wish to delete and confirm it on the subsequent page.

Note: Trap Console does not offer the Edit and Delete hyperlinks for the administrator. It is not possible to delete the administrator account or change his/her access rights.

Licensing

The first tab on the **Preferences** page contains information about the number of licenses for Trap Console. Trap Console is licensed per number of unique SNMP agents (measured by unique source and agent addresses within incoming SNMP traps). Provided that Trap Console has processed SNMP traps from more agents than the actual number of licenses, a warning with explanation appears (see [Figure 54](#)). In this case you should obtain a sufficient amount of licenses to allow Trap Console processing all incoming traps.

Preferences

Licensing	Application Log	Address Restrictions	Restricted Trap Log	Mapping	Vari
-----------	-----------------	----------------------	---------------------	---------	------

[Licensing](#)

Trap Console is licensed per number of unique SNMP agents (measured by unique source and agent addresses within incoming SNMP traps).

License: 5 SNMP sender or agent addresses

WARNING: Trap Console processed SNMP traps from 6 different SNMP sender or agent addresses (see the [log file](#) for details). You must obtain at least 1 additional Trap Console licenses.

[SNMP Agent Statistics](#)

[License Key Manager](#)

Trap Console's license keys. For security reasons, serial numbers instead of actual license keys are being displayed.

Please, contact <http://www.cscare.com/trapconsole> to obtain licenses.

New License Key:

Key Serial Number	Number Of Licenses	
085-10345778	5	<input type="button" value="Remove"/>
Total: 5		

Figure 54, Licensing Information

Note: License key manager is available only in the commercial version of Trap Console. Thus, it is not possible to add license keys to the evaluation version of Trap Console.

In the **License Key Manager** section enter the new licence key into the provided edit box and press **Add**. In the table below a list of your licence keys is displayed. It is possible to remove a licence key clicking on the respective **Remove** link.

Agent Statistics

To see trap statistics, follow the **SNMP Agent Statistics** link (see [Figure 54](#)). **SNMP Agent Statistics** page is accessible also from Preferences menu (tabs) It shows vital information about incoming traps (see [Figure 55](#)).

Runtime

Total Number of Processed Traps: 11

UDP Trap Receiver	TCP Trap Receiver	Syslog UDP Receiver	Syslog TCP Receiver	SNMP Packet Log	SNMP Agent Statistics
-------------------	-------------------	---------------------	---------------------	-----------------	-----------------------

This page displays all Trap Agents and some statistical data: the count of traps from each agent, the last and average interval between two Traps and the time of the last Trap.
Use the **Refresh** button to actualize the table of statistics.

SNMP Agent Statistics are enabled: **Disable Statistics**

Refresh **Reset**

Note:

Format of the date and time displayed in column "Last Trap" is: YYYY.MM.DD HH:MM:SS

***** License violation - number of trap sender or agent addresses over the license: 1; total licenses needed: 6 *****

Sender Address	Agent Address	Traps	Licenses	Average Interval	Last Interval	Last Trap
127.0.0.1	192.168.1.6	4	2	42m 39s	2 h 7 m 28 s	2005.07.20 17:48:30
127.0.0.1	192.168.1.1	2	1	13m 7s	13m 7s	2005.07.20 18:01:46
127.0.0.1	192.168.1.2	1	1			2005.07.20 17:48:46
127.0.0.1	192.168.1.5	1	1			2005.07.20 17:54:12
127.0.0.1	192.168.1.3	1	1			2005.07.20 17:54:22
127.0.0.1	192.168.1.7	1	1			2005.07.20 17:54:33
127.0.0.1	192.168.1.8	1	1			2005.07.20 18:01:32

Figure 55, Agent Statistics

Each row shows the sender and agent address, the number of traps received, the number of licenses used, and the time of the last trap received. If more than one trap has been received from one address, then the average and the last interval is calculated and displayed in seconds.

If traps arrive from more agents than there are licenses available, then a **License violation** information appears, stating the number of trap agents over license and the total number of licenses needed. Traps received from agents over license are displayed in red.

To refresh the statistics, press the **Refresh** button, or if you wish to start the statistics from now on, press the **Reset** button.

Note: After resetting the statistics, the previous information cannot be retrieved.

Tuning Trap Console

The **Runtime** page groups configuration, statistics and other dynamic control concerning trap processing. There is a separate page for each of Trap Console's receivers, SNMP trap/notification receivers for UDP and TCP transport and syslog receivers for UDP and TCP. Receiver control pages have identical layout, see [Figure 56](#), [UDP Trap Receiver](#) as an example.

Receiver can be started/stopped on-the-fly, using **Start Receiver/Stop Receiver** button. Binding to a specific local port is settable in **Receiver Port** edit line, pressing **Set** will persistently set the port number. If running, the receiver will be restarted automatically.

Next line shows current number of processed traps, use **Refresh** to refresh the counter. **Reset** button clears the counter. Number of processed traps is present on every receiver page and represents number of successfully parsed traps, not necessarily related to the number of packets, connections or syslog messages.

Runtime

Total Number of Processed Traps: 11

UDP Trap Receiver	TCP Trap Receiver	Syslog UDP Receiver	Syslog TCP Receiver	SNMP Pac
-------------------	-------------------	---------------------	---------------------	----------

UDP Trap Receiver

UDP Trap Receiver is started

Receiver port:

Number of Processed Traps: 11

Last reset on July 20, 2005 1:06:25 PM CEST

UDP Packet Queue

Received/Dropped Packets: 11 / 0

Buffered Packets: 0

Maximum Number of Buffered Packets:

Number of Threads [All/Waiting]: 4 / 4

Maximum Number of Threads:

Figure 56, Performance Tuning

UDP Packet Queue section contains trap packet queue statistics. Packets are received and put into the queue. Dropping the packet occurs when the queue is full. **Buffered packets** is the number of packets waiting in the queue for processing thread to become available.

Note: If this number grows or stays nonzero for a longer time while the count of waiting threads is near to zero value, consider raising the **Maximum Number of Threads** value (explained further).

Maximum Number of Buffered Packets is set to 256 by default. This is the maximum number of request the queue is capable to hold. If you wish to change his value, enter the new value in the edit box and press **Set**.

Note: Once the count of buffered packets reaches the **Maximum Number of Buffered Packets** value, Trap Console starts dropping subsequent requests. In order to prevent such a situation, consider increasing the maximum number of buffered packets value as well as the maximum number of threads value.

Number of Threads row displays the number of **all** threads available for trap processing for the particular protocol and the number of threads **waiting** idle for incoming requests.

Maximum Number of Threads displays the maximum number of threads that will be available to process incoming requests. The default value is 32 threads. It can be changed by entering a value in the edit box and pressing **Set**.

Note: Increasing the amount of threads results in more resources given to Trap Console - it must be handled carefully so as not to clog the computer. Generally, Trap Console takes care that it runs within the resources it was assigned to and that it will not bring the system down, even during the highest loads.

When a packet arrives, Trap Console hands it to one of its processing threads for processing. Trap Console creates threads on the 'as needed' basis until it reaches the **Maximum Number of Threads** count. When no thread is available and the limit is reached, Trap Console places the packet the queue. As soon as a thread finishes processing the packet, it retrieves the first waiting packet from the queue, or when no packet is available, falls asleep and waits for another. During a lower load, Trap Console keeps only the half of **Maximum Number of Threads** waiting, destroying the others in order to preserve system resources.

Changing the **Maximum Number of Threads** value gives Trap Console more power to handle higher traffic, while consuming more system resources. Changing this value optimizes the performance of Trap Console with respect to the system resource utilization.

Working with the Application Log File

The application log file `trapconsole.log` is created during the installation.

This file contains application-related events that occur from the point of starting Trap Console (or from deleting the log file). Those events include user logins, logouts, the Trap Console starts, stops, authentication failure, error reports and more.

When testing an action, view the application log file for related events. Also, Trap Console writes here traps received from agents with IP addresses over the license.

The following buttons (see [Figure 57](#)) appear when you select the **Application Log** tab on the **Preferences** page.

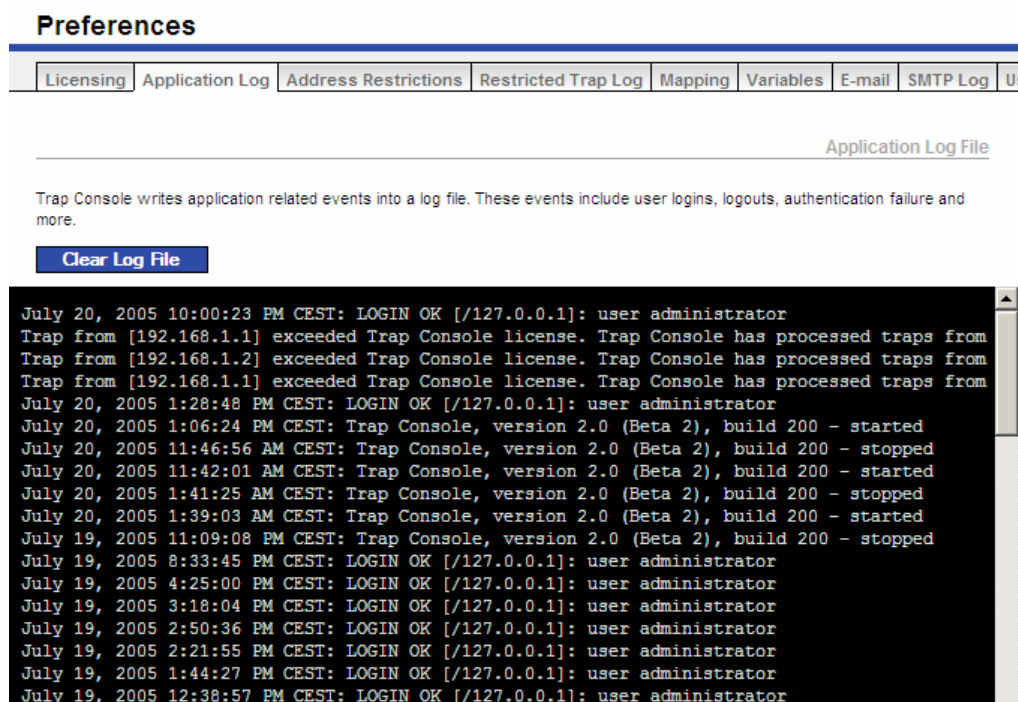


Figure 57, Working with Application Log File

To clear the application log file, click the **Clear Log File** button.

E-mail Settings

This part of the **Preferences** page allows you to set global e-mail parameters (see [Figure 58](#)).

There are two kinds of global e-mail parameters. The first one (SMTP Server) is used for all outgoing messages and the second one is designed for the case of license overflow.

SMTP Server is the IP address/name of the mailserver Trap Console uses to send all e-mail messages out. Trap Console sends e-mail messages as a result of user defined actions (see the section '[Creating a New E-mail Action](#)' on page 44) or during the license overflow. In both cases the same SMTP server is used.

You can specify **Multiple SMTP servers** (separated with semicolon) to back the SMTP server in case of failure of the first SMTP server.

Sender (From) is an e-mail address Trap Console uses as the sender of e-mails.

Preferences

Licensing Application Log Address Restrictions Restricted Trap Log Mapping Variables E-mail SMTP Log

E-mail Settings

SMTP Server: localhost

Sender (From): trapconsole@localhost

Administrator (To): admin@cscare.com

L/O Notifications:

Message Spool Directory: spool

Message Spool Scan Interval: 10000 [ms]

Logging Level: 2

Log File Size Limit: 100 [kB]

Maximum Sender Threads: 1

SMTP Socket Timeout: 300000 [ms]

Save Changes

Figure 58, Global E-mail Settings

Administrator (To) is an e-mail address of this Trap Console license administrator. Trap Console will send her/him notification messages whenever the license overflow occurs in case the notification sending is enabled.

L/O (License Overflow) Notifications enables/disables sending license overflow notification messages via e-mail to the administrator's address given here. A notification message will be sent only once a day even if the license overflow occurs more times during the day (see the section '[License Overflow](#)' on page 74).

Message Spool Directory is an absolute or relative path (relative to Trap Console installation directory). This directory will temporarily contain email messages created by Trap Console actions before they get forwarded to **SMTP Server**.

Message Spool Scan Interval is a time interval when the message spool processing occurs.

Logging Level is the current severity level of log messages. Log messages of severity equal and higher than this level will be included in the log.

Log File Size Limit is the maximum size of individual log files. When the size of a log file reaches this limit, the logging will continue writing in a new file or rewriting the last used file.

Maximum Sender Threads is the maximum number of SMTP sender threads that will simultaneously forward messages found in the message spool directory to SMTP Server.

SMTP Socket Timeout is the timeout of the SMTP socket while communicating with the SMTP server. Inactivity longer than this value will cause closing the communication channel.

Do not forget to commit changes by clicking the **Save Changes** button.

After the installation, Trap Console creates the subdirectory `spool` where e-mail messages are spooled before sent out. All messages are sent out from the spool in the pre-defined period. The subdirectory name and the period can be adjusted in the `trapconsole.cfg` file (see [Customizing trapconsole.cfg](#)).

SMTP Log

The SMTP log file contains a trace of mailing activities of Trap Console. The default log file name is `smtpsender.log` and its maximal size is adjusted to 1000 KB by default. You can change both parameters in the `trapconsole.cfg` file, too. Even you can influence the level of logged information there. We recommend increasing the level of logged information for troubleshooting purposes only.

To clear the SMTP log file, click the **Clear SMTP Log File** button.

Incoming Trap Restrictions

It is possible to restrict IP addresses from which Trap Console accepts SNMP traps. Restrictions apply to both sender and agent addresses.

Select the **Preferences** page and click on the **Address Restrictions** link. The following page appears:

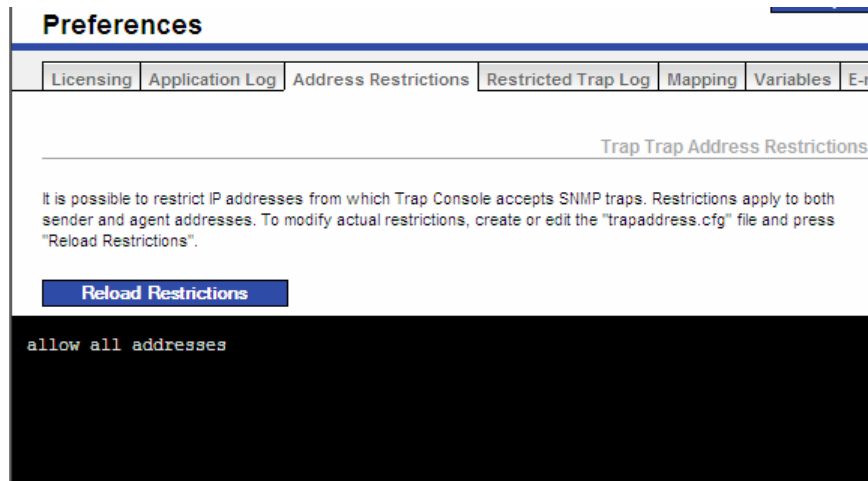


Figure 59, Trap Address Restrictions

The **Address Restrictions** page lists applied restrictions. To modify actual restrictions, create or **edit** the `trapaddress.cfg` file and press **Reload Restrictions**.

Editing trapaddress.cfg

`trapaddress.cfg` is a text file. Each line defines one address restriction rule.

The syntax is:

```
[ { "allow" | "deny" } ] { "*" | | }
```

which means that there is a possibility to have rules for all addresses ("*"), one IP address or for IP address range. Access may be either allowed via "allow" or denied via "deny".

Empty lines and lines starting with "#" are ignored.

If the file does not exist or in case it contains no rule, all IP addresses are accepted. It is possible to use host names.

Example:

```
allow 10.0.0.0..10.0.0.100
allow 127.0.0.1
deny *
```

Explicitly denies access from any IP address with the exception of range 10.0.0.0 .. 10.0.0.100 and 127.0.0.1.

The order of rules is important – the first matching rule prevails.

Logging of Traps Discarded by Address Restrictions

Information about discarded traps can be recorded in a log file. To do so, tick the **Enable** checkbox, and set the **Log File Size Limit** in kB (100 kB is set by default) (see Figure 60).

In the **Log Line Template**, specify the text line that will be recorded in the log file upon discarding a trap from the restricted address. The provided text box can contain macros. For detailed description of macros, see [Macros](#)

on page 86. To apply changes, press **Save Changes**.

The log file can be deleted by pressing the **Clear Log File** button and confirming the intention to delete it by confirming it on the subsequent page.

Preferences

Licensing	Application Log	Address Restrictions	Restricted Trap Log	Mapping	Variables	E-mail	SMTP Log
-----------	-----------------	----------------------	---------------------	---------	-----------	--------	----------

Address Restricted Trap Log File

Logging of traps discarded by Address Restrictions.

Information about discarded traps can be recorded in log file.

Enable:

Log File Size Limit: [kB]

Log Line Template:

```

July 20, 2005 11:12:11 PM CEST: snmp/linkUp from 127.0.0.1/localhost for 192.168.1.6/echos
July 20, 2005 11:11:59 PM CEST: Log was cleared.

```

Figure 60, Address Restrictions Log

Web Browser Session Timeout

Session timeout is a time interval used by the server before automatically disconnecting an inactive user session. For some installations it is advantageous to raise the session timeout if there are only a few users logged in for long periods of time with little activity.

Clicking on the **Timeout** link on the **Preferences** page and enter the preferred session timeout in minutes in the provided edit line and press **Set**.

Trap Variable to Environment Variable Mapping

A received trap can launch (if set) an external application. A simple way how to let the application know about trap variables is to pass the values into application process environment.

On the **Preferences** page, **Mapping** tab, the displayed table (see [Figure 61](#)) lists macro names and their corresponding environment variable name mappings. The environment variable contains a value of the trap variable represented by a macro name.

Preferences

Licensing	Application Log	Address Restrictions	Restricted Trap Log	Mapping	Variables	E-mail	SMTP Log	User Manager	Passw
-----------	-----------------	----------------------	---------------------	---------	-----------	--------	----------	--------------	-------

Trap Variable to Environment Variable Mapping

A received trap can launch (if set) an external application. A simple way how to let the application know about trap variables is to pass the values into application process environment. The following table lists macro names and their corresponding environment variable name mappings. Environment variable contains a value of the trap variable represented by a macro name.

Trap Variable	Environment Variable			
<input type="text"/>	<input type="text"/>	<input type="button" value="Add/Update"/>	<input type="button" value="Remove"/>	<input type="button" value="Reload Defaults"/>
agentAddress	TRAPCONSOLE_AGENTADDRESS			
agentName	TRAPCONSOLE_AGENTNAME			
amPm	TRAPCONSOLE_AMPm			
community	TRAPCONSOLE_COMMUNITY			
day	TRAPCONSOLE_DAY			
description	TRAPCONSOLE_DESCRIPTION			
encoding	TRAPCONSOLE_ENCODING			
enterprise	TRAPCONSOLE_ENTERPRISE			
enterpriseId	TRAPCONSOLE_ENTERPRISEID			

Figure 61, Trap Variable to Environment Variable Mapping

In order to add or update a variable mapping, type the trap variable name in the **Trap Variable** edit line or select an existing variable mapping by clicking its trap variable name (hyperlink) in the list below. Edit the environment variable name in the **Environment Variable** edit line and press the **Add/Update**.

To remove an existing variable mapping, Select a variable mapping by clicking its trap variable name (hyperlink) in the list and press **Remove** button.

Press **Reload Defaults**, after deleting or adding a large number of mappings, to restore the default trap variable mappings.

Note: Trap Console environment variables are included in the `envvars.cfg` file.

Environment for External Applications

External applications launched by Trap Console may require certain environment variables to be set appropriately. The displayed table (see Figure 62) shows variable definitions that the application environment contains.

Preferences

Licensing	Application Log	Address Restrictions	Restricted Trap Log	Mapping	Variables	E-mail	SMTP Log	User Mana
-----------	-----------------	----------------------	---------------------	---------	-----------	--------	----------	-----------

Environment for External Ap

External applications launched by TrapConsole may require certain environment variables to be set appropriately. The displayed lists show variable definitions that the application environment will contain. User defined environment variables at the top of the list take precedence over system envii variable values. System environment variables below mirror the system environment at the time of the TrapConsole setup or the time they were rel

Variable Name	Variable Value			
<input type="text" value="CLIENTNAME"/>	<input type="text" value="Console"/>	<input type="button" value="Add/Modify"/>	<input type="button" value="Remove"/>	<input type="button" value="Reload"/>
User defined environment variables				
ALLUSERSPROFILE				
CLIENTNAME	Console2			
SESSIONNAME	Console2			
System environment variables				
(removed) ALLUSERSPROFILE	C:\Documents and Settings\All Users			
CLASSPATH	.;C:\Program Files\Java\jre1.5.0_06\lib\ext\QTJava.zip			
(redefined) CLIENTNAME	Console			
COMPUTERNAME	CSDEV			

Figure 62, Environment for External Applications

Type the variable name in the **Variable Name** edit line or select an existing variable definition by clicking its name (hyperlink) in the lists below. Edit the variable value in the **Variable Value** edit line and press the **Add/Update** button

To remove an existing variable mapping, select it by clicking its name (hyperlink) and press **Remove**.

Note: Environment settings are already set provided Trap Console has been installed using the installation package for MS Windows systems. If the **zip package** has been used for installation, environment variables will be set after running the `envdump.exe` file (creating the `envset.cfg`) in Trap Console's working directory and restarting Trap Console. In case of running Trap Console on Linux systems, environment variables have to be entered manually or you can dump the environment using command like "env > envset.cfg" and verify it loads correctly.

Console Command Interface

Clicking on the **Console** link on the **Runtime** page, the **Console Command Interface** page appears.

This page can be used as a substitution for the console window. The advantage is that the console commands can be performed remotely using the Web browser. Enter any of the console commands listed on the page in the command line and press **Submit**.

Please be cautious when applying the 'q' command. Trap Console will stop afterwards, however, it is not possible to re-start Trap Console via Web browser.

Note: Console command interface is available only for the administrator.

Working with Databases

In the section 'Writing Traps to the Database' on page 42 you have been introduced to writing traps into databases briefly. Since Trap Console enables to write traps to any JDBC compliant database, we will look at this database type access in more details. This section guides you through the basics of JDBC, JDBC drivers and shows you examples of using them.

What is JDBC?

Java Database Connectivity (JDBC™) is a standard vendor-independent Java interface for connecting to relational databases. It provides uniform access to a wide range of relational databases. JDBC also provides a common base on which higher level tools and interfaces can be built. JDBC is the mechanism for Java applications (as Trap Console is) to talk to different databases on different platforms.

To use JDBC with a database management system, you need a JDBC Driver to mediate a connection between JDBC and the database. A JDBC Driver is platform-specific.

A Java application can use two JDBC driver types for access to a database:

(In fact, there are four JDBC driver types, but it is not the subject of this User's Guide).

- JDBC-ODBC Bridge plus ODBC driver - it enables a JDBC access via existing ODBC drivers to any ODBC database.
- Native Driver – it converts JDBC calls into calls on the client API for Oracle, Sybase, Informix or other databases on different platforms.

Microsoft's ODBC (Open Database Connectivity) API is probably the most widely used programming interface for accessing relational databases on the Microsoft Windows platform. The JDBC-ODBC Bridge is a JDBC driver, which implements JDBC operations by translating them into ODBC operations. To ODBC it appears as a normal application program. The Bridge implements JDBC for any database for which an ODBC driver is available. The JDBC-ODBC bridge is implemented by the following Java package:

- `sun.jdbc.odbc` in Java™ Development Kit

Trap Console detects automatically the JVM type and suggests the correct driver.

Connecting to different databases on Microsoft Windows platform might be possible either through the JDBC-ODBC bridge driver or through a native driver. The second method can be used on any platform, including Microsoft Windows.

Native JDBC drivers are usually developed by the database vendors or by third-party driver developers. Often a native JDBC driver is a part of the database installation package.

For more information on using JDBC-ODBC bridge driver see the following sections:

- '[How to Write Traps to a Microsoft Access Database](#)' on page 65 or
- '[How to Write Traps to an Oracle Database through JDBC-ODBC Bridge](#)' on page 68.

For more information on using native JDBC drivers, see the section '[How to Write Traps to an Oracle Database through Native Drivers](#)' on page 70.

Information in this section may quickly become outdated, so we suggest you to consult the following JDBC Web page

<http://java.sun.com/products/jdbc>

How to Write Traps to a Microsoft Access Database

If Trap Console should write trap information to a Microsoft Access database, use the JDBC-ODBC Bridge. Follow these steps:

- Create a new Microsoft Access database,
- Create a table with required items
You can use an existing database and create only the table.
- Create a new ODBC data source by using the ODBC Administrator.
A data source specifies the Microsoft Access database or the table in the database you want to use and information needed to get to that database. A data source is identified by the Data Source Name (DSN). The ODBC Administrator asks for the driver-specific information. In case

- of the Microsoft Access Driver you need only to locate the Microsoft Access database you want to access.
- Open the Create New Action page in Trap Console and create a new action with the action type 'Write trap to the database'. Enter attributes of the new action:
 - *The action name*
 - *JDBC driver name* – Trap Console writes the correct JDBC-ODBC bridge into the JDBC driver name as follows:

```
sun.jdbc.odbc.JdbcOdbcDriver
```

 - if you use JavaSoft's JDK/JRE
 - *Database URL* – for Microsoft Access database use the following:

```
jdbc:odbc:<subname>
```

where subname is the DSN registered with ODBC Administrator.
 - *SQL statement* – statement to be performed.
Microsoft Access INSERT INTO statement has the following syntax:

```
INSERT INTO tablename [(itemname1[, itemname2[, ...]])]  
VALUES (value1[, value2[, ...]);
```

where

tablename – is the name of the created table

itemnamex – the name of the x-th table field

valuex - the value to insert into the x-th field of the new record.

Note: If Trap Console runs as an NT service and it needs to use the JDBC-ODBC bridge, you must change the Trap Console log-on account. Trap Console (as an NT service) uses the system account by default. However the JDBC-ODBC bridge requires to use a user account. Therefore you need to change the Trap Console working account. To change the account, open the '**Services**' applet and double-click on '**CSCare Trap Console**' in the list of services. The **Service Properties** dialog window appears, and there, select the **Log On** tab (see [Figure 63](#)). Tick the **This account** radio box, press the **Browse** button, select the appropriate account from the listed ones, and press OK.

You can use a password if you wish, but then you need to enter the password in Trap Console to get access to database (see [Figure 64](#)).

If you do not want to use a password, clear the password field.

Finally, press the **Apply** button in the right bottom corner of the dialog window to set the account.

Do not forget to check the access rights of the selected account.

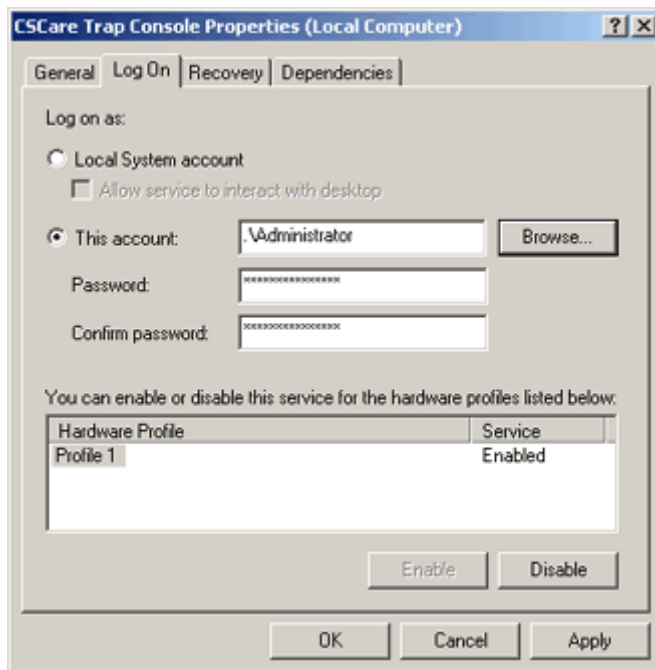


Figure 63, Setting Trap Console Account

Example:

Let us to show you how to write SNMP trap information to the Microsoft® Access2000 database in Microsoft Windows 2000.

The database Traps.mdb, created in Microsoft® Access, contains the table SNMPTrapsAccess with the following text items (of length 100):

- TrapName – the name of the trap,
- HostAddress - IP address of the SNMP agent which sent the trap,
- HostName - the host name of the SNMP agent which sent the trap,
- Description - the description of the trap.

To access this database from Trap Console, create a new Data Source Name (DSN) item SNMPTrapsAccess in ODBC Administrator. Connect it to the Traps.mdb file.

Trap Console will use this DSN 'SNMPTrapsAccess' in the **Database URL**.

This example has been tested in the following environment:

- Microsoft Windows 98, 2000,
- JDBC-ODBC Bridge Driver from JDK version 1.1.8,
- Microsoft Access 97, 2000,
- Microsoft Access ODBC Driver and,
- JRE from JDK version 1.1.8.

Now we can create a new action 'Write to my database SNMPTraps'. In [Figure 64](#) you can see attributes of the action:

Upon reception of an SNMP trap, this action executes the specified SQL command in the selected database.

Action Name:

JDBC Driver:

Database URL:

User:

Password:

SQL Statement:

Conn. Min.: [sec]

Conn. Max.: [sec]

Timeout: [sec]

Figure 64, Writing Traps to a Microsoft Access Database

- If you set a password for the Trap Console log-on account, then fill in the **User** name and **Password** in the provided edit boxes.
- We recommend you to test the created action using the **Test Action**. You can simulate an arbitrary trap and try to write it to the database. This way you can discover and fix potential problems now, in the action design phase.

How to Write Traps to an Oracle Database through JDBC-ODBC Bridge

If you run Trap Console on a Microsoft Windows machine and you want to write traps to Oracle database, you can use the JDBC-ODBC bridge driver. Note that an Oracle client must be present in this case.

Follow these steps:

- Prepare a table in the Oracle database.
- Create a new ODBC data source by using the ODBC Administrator.

The data source determines the table in the Oracle database you want to use and information needed to input. The data source is identified by the Data Source Name (DSN). ODBC Administrator asks for the driver-specific information. In case of the Driver for Oracle you need to specify the following items:

 - New DSN – the DSN for the prepared table
 - User Name – for access to the database
 - Server – the database server, on which the database works. If you don not know the correct name, consult your database administrator.
- Open the Create New Action page in Trap Console and create a new action with the action type 'Write trap to the database' and fill all the necessary items for the new action such as :
 - *The action name*
 - *JDBC driver name* – Trap Console writes the correct JDBC-ODBC bridge into the JDBC driver name as follows :


```
sun.jdbc.odbc.JdbcOdbcDriver
```

 - if you use JavaSoft's JDK/JRE
 - *Database URL* – for access to the Oracle database through ODBC driver use the following:


```
jdbc:odbc:<subname>
```

, where

subname is the DSN registered in Microsoft Windows.

- *Username* – your User Name for access to the Oracle database. If you leave it empty, Trap Console takes the name specified in ODBC Administrator.
- *Password* – your password for access to the Oracle database. This item is required. You cannot access any Oracle database without the correct password.
- *SQL statements* – statements to be performed.

```
INSERT INTO username.tablename [(itemname1[, itemname2[, ...]])]
```

```
VALUES (value1[, value2[, ...])
```

where

username – your user name for access to the table

tablename – is the name of the created table

itemnamex – the name of the x-th table field

valuex – the value to insert into the x-th field of the new record.

Note: If Trap Console runs as an NT service and it needs to use the JDBC-ODBC bridge, you must change the Trap Console log-on account. Trap Console (as an NT service) uses the system account by default. However the JDBC-ODBC bridge requires to use a user account. Therefore you need to change the Trap Console working account. To change the account, open the '**Services**' applet and double-click on '**CSCare Trap Console**' in the list of services. The **Service Properties** dialog window appears, and there, select the **Log On** tab (see [Figure 63](#)). Tick the **This account** radio box, press the **Browse** button, select the appropriate account from the listed ones, and press OK.

You can use a password if you wish, but then you need to enter the password in Trap Console to get access to database (see [Figure 64](#)).

If you do not want to use a password, clear the password field.

Finally, press the **Apply** button in the right bottom corner of the dialog window to set the account.

Do not forget to check the access rights of the selected account.

Example:

In the following example we use the table with text items (of length 100):

TrapName - the name of the trap,

HostAddress - IP address of the SNMP agent which sent the trap,

HostName - the host name of the SNMP agent which sent the trap,

Description - the description of the trap.

This example has been tested in the following environments:

- Microsoft Windows NT 4.0 Workstation, Windows 2000,
- Microsoft ODBC Driver for Oracle,
- JDBC-ODBC Bridge Driver from JDK version 1.1.8,
- Oracle database version 8.1.5 and,
- JRE from JDK version 1.1.8.

[Figure 65](#) shows how create the action 'Write to Oracle database through JDBC-ODBC bridge'.

Upon reception of an SNMP trap, this action executes the specified SQL command in the selected database.

Action Name:

JDBC Driver:

Database URL:

User:

Password:

SQL Statement:

Conn. Min.: [sec]

Conn. Max.: [sec]

Timeout: [sec]

Figure 65, Writing Traps to an Oracle Database JDBC-ODBC Bridge

- If you set a password for the Trap Console log-on account, then fill in the **User** name and **Password** in the provided edit boxes.
- We recommend you to test the created action using the **Test Action** button. You can simulate an arbitrary trap and try to write it to the database. This way you can discover and fix potential problems now, in the action design phase.

How to Write Traps to an Oracle Database through Native Drivers

This section describes the existing types of Oracle JDBC Drivers and shows how to use them correctly.

Driver Types

At the time of this writing, Oracle offers two different JDBC drivers for the client-side use:

- **JDBC Thin Driver** – is a pure Java implementation and is platform-independent. It does not require any Oracle software on the client side. The JDBC Thin driver connects to any Oracle database of version 7.2.3 and higher.
- **JDBC OCI Driver** – provides an implementation of the JDBC interfaces using the Oracle Call Interface (OCI). The use of OCI makes the driver platform specific. It requires the Oracle client installation. The JDBC OCI Driver is compatible with all Oracle versions.

If you need maximum portability or need to access the Oracle database from a host without the Oracle client software, choose the JDBC Thin Driver.

If you need maximum performance, use the JDBC OCI driver.

How to Use Native Oracle JDBC Drivers

To write traps into Oracle database through native JDBC driver, follow these steps:

- Install the Oracle client software with JDBC drivers to your computer.
- Add driver package to the CLASSPATH
- Prepare a table in the database.
- Create a new Trap Console action using the native JDBC driver.

The following paragraphs describe it in more details.

Installation of the correct Oracle JDBC driver

Installation of Oracle JDBC drivers is platform-specific. Follow the instructions in the Oracle driver documentation.

Note that if you install the JDBC OCI driver, you must install also the Oracle client software.

You can download the Oracle JDBC drivers' latest version from

<http://www.oracle.com/technology/software/index.html>.

Adding driver package to the CLASSPATH

When starting Trap Console, the appropriate Oracle JDBC driver package must be added to the CLASSPATH.

According to Java version, use one of following packages:

For use with JDK 1.4

`ojdbc14.jar` - JDBC classes

`ojdbc14_g.jar` - JDBC classes with debug and trace

On windows platform you need to change the trapconsole.bat file as follows:

```
@echo off
REM The batch file for launching the Trap Console server

.\jre\bin\java -server -cp ojdbc14.jar;.tc.jar TrapConsole.Main
```

Preparing a table

You need to prepare a table with required trap information in the Oracle database.

Creating a new Trap Console action using the native JDBC driver

Open the Create New Action page in Trap Console and create a new action with the action type 'Write trap to the database' and fill all the necessary items for the new action such as:

- The action name
- *JDBC driver name* – for both driver types, write it as follows:
oracle.jdbc.driver.OracleDriver
- *Database URL* – for access to the Oracle database through native JDBC drivers use a database URL in the form:
jdbc:oracle:<drivertype>:@<database>, where
drivertype is the type of the used driver:
thin - for the JDBC Thin Driver or
ocix - for the JDBC OCI Driver, where x represents the Oracle driver version (at the time of this writing: oci7 or oci8. It depends on the used Oracle database version).
database identifies the database to which you want to connect. This part of the database URL depends on the Oracle JDBC driver type.
- *Username* – your User Name for access to the Oracle database. This item is required.
- *Password* – your password for access to the Oracle database. This item is required. You cannot access any Oracle database without the correct password.
- *SQL statements* – statements to be performed.
INSERT INTO username.tablename [(itemname1[, itemname2[, ...]])]
VALUES (value1[, value2[, ...])
where
username – your user name for access to the table

tablename – is the name of the created table
itemnamex – the name of the x-th table field
valuex – the value to insert into the x-th field of the new record.

Examples:

The steps in the previous section are illustrated in two following examples. The first example shows how to use the Oracle JDBC Thin driver (see [Figure 66](#)). The second one illustrates using the Oracle JDBC OCI driver (see [Figure 67](#)).

This section assumes you have already installed JDBC Drivers and set the CLASSPATH according to the used JDK version:

We write to the table with the following items (of length 100):

TrapName – the name of the trap,
HostAddress - IP address of the SNMP agent which sent the trap,
HostName - the host name of the SNMP agent which sent the trap,
Description - the description of the trap.

Using Oracle JDBC Thin Driver

[Figure 66](#) illustrates the Trap Console action which uses the Oracle JDBC Thin Driver to access the Oracle database.

Note that the database URL part @<database> must contain:

- explicitly the name of the host, on which the Oracle server works
- TCP/IP port
- Oracle SID (system identifier).

If you do not know it, please contact your database administrator.

In the following example, this database URL is used:

```
jdbc:oracle:thin:@dbserver:1521:orc1.
```

It means that Trap Console will write trap information to the database on the host **dbserver**. The server dbserver has a TCP/IP listener on the port **1521** for the database SID **orc1**.

The example was tested in this environment:

- Microsoft Windows NT, 2000,
- Oracle JDBC Thin Driver version 8.1.5,
- Oracle database version 8.1.5 on the host and
- Java virtual machine JRE from JDK version 1.1.8.

Note: If you use this driver, it is not necessary to install any Oracle client software. You must install only the driver.

Upon reception of an SNMP trap, this action executes the specified SQL command in the selected database.

Action Name:

JDBC Driver:

Database URL:

User:

Password:

SQL Statement:

Conn. Min.: [sec]

Conn. Max.: [sec]

Timeout: [sec]

Figure 66, Writing Traps to Oracle Database through Oracle JDBC Thin Driver

Using Oracle JDBC OCI Driver

Figure 67 illustrates the Trap Console action, which uses the Oracle JDBC OCI Driver to access the Oracle database.

The database URL part @<database> contains the name of the Oracle server. If you do not know it, contact your database administrator.

The example was tested in this environment:

- Microsoft Windows NT, 2000,
- Oracle JDBC OCI Driver version 8,
- Oracle database version 8.1.5 and
- Java virtual machine JRE from JDK version 1.1.8.

Upon reception of an SNMP trap, this action executes the specified SQL command in the selected database.

Action Name:

JDBC Driver:

Database URL:

User:

Password:

SQL Statement:

Conn. Min.: [sec]

Conn. Max.: [sec]

Timeout: [sec]

Figure 67, Writing to Oracle Database through Oracle JDBC OCI Driver

We recommend you to test the created action using the **Test Action** button. You can simulate an arbitrary trap and try to write it to the database. This way you can discover and fix potential problems now, in the action design phase.

License Overflow

Trap Console is licensed per number of SNMP agents it handles. If you have licensed Trap Console for 10 SNMP agents, Trap Console handles SNMP traps being received from up to 10 different SNMP agents (IP addresses). Trap Console records IP addresses from the 'agent address' field of each trap it receives. The number of unique agent IP addresses must be less than or equal to the number of Trap Console licenses (i.e. 10 in this example). All traps received from additional IP addresses are indicated as License Overflow.

Traps over the license were just logged into the application log file and discarded in Trap Console version less than 1.3. This caused problems if new SNMP agent started to send traps to Trap Console. Its traps were discarded unless you have added new licenses. In such case you could not control this SNMP agent by means of Trap Console.

Starting from the version 1.3 such traps are handled as those under the license. In addition, Trap Console executes several notification procedures to warn the administrator that the license is exceeded.

Trap Console provides the following notification procedures:

- Writes traps over the license to the application log file.
- Writes the warning into the Trap Console' console window.
- Writes the warning on Web browser page.
- Sends a notification e-mail message to the Trap Console administrator (only if this is allowed on the **Preferences/E-mail** page).

The e-mail notification will be sent to the administrator address, specified on the **E-mail** page (see the section 'E-mail Settings' on page 59).

The administrator will find the following information in the message:

- The IP address/name of the SNMP agent which sent the trap.
- The Trap Console version.
- The IP address/name of the host, on which Trap Console works.
- The number of the Trap Console licenses.
- The number of license overflows.
- The contact where additional licenses could be obtained.

This notification message will be sent whenever the license overflow occurs, but not more often than once a day.

On the **Runtime** page in the **SNMP Agent Statistics** tab, there is the **License Overflow** counter, which shows how many license overflows have occurred from the Trap Console start.

Need Help?

Here are some suggestions how to obtain more information on Trap Console:

- See the `readme.txt` file
- See the **Description** link next to the page heading for more detailed information (if available).
- Invoke the Trap Console Web browser manager **Help** page (see [Figure 68](#)) containing the most discussed topics.

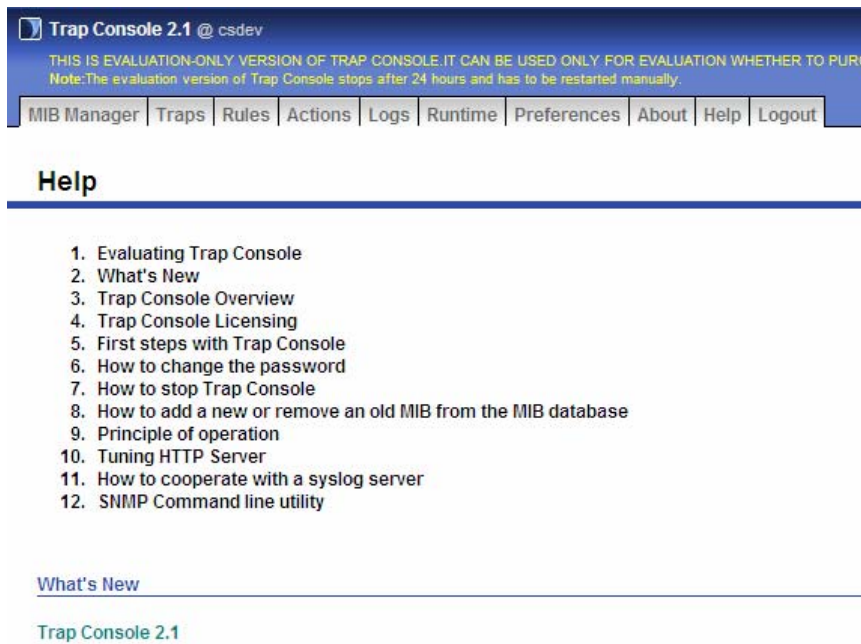


Figure 68, Help Page

- Invoke the **About** page to get information about Trap Console Version, License and Contact Information.
- See the Trap Console news at <http://www.cscare.com/trapconsole>
- Contact us via e-mail on trapconsole@cscare.com.

SNMP Primer

What is SNMP

SNMP (Simple Network Management Protocol) is a protocol designed to give the user the capability to remotely manage a computer network by selecting and setting terminal values and monitoring network events. SNMP was designed to enable communication between different types of network and allow different types and brands of network peripherals (hubs, bridges, routers, etc) to be managed by a single piece of network management software.

SNMP is used to exchange information between network management stations and agents in network elements. Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, gateways, terminal servers, etc., which have management agents responsible for performing the network management functions requested by network management stations.

The Advantages of SNMP

- The greatest advantage of using SNMP is that its design is simple, thus the development cost for management software is accordingly reduced.
- An extensibility of SNMP to accommodate additional, possibly unanticipated aspects of network operation and management.
- An architecture which is, as much as possible, independent of the architecture and mechanisms of particular hosts or particular gateways.
- SNMP gives the user the capability to remotely manage a computer network by receiving and setting terminal values and monitoring network events.
- SNMP is widely used today. The result of this is that almost all major vendors of internetwork hardware, such as bridges and routers, design their products to support SNMP, making it very easy to implement.

The Components of SNMP

SNMP is composed of three basic elements:

- The Manager,
- The Agent, and
- The MIB (Management Information Base).

The Manager

The manager is an application that runs on the host computer. It provides the interface through which a human manager (or a program) performs management activities. Its main role is to select agents for certain requested information.

Many manager programs and several tools to build these programs exist in the computer world. CSCare Inc., the Trap Console developing company, also ships one of them - the Visual SNMP - which is a visual developing tool for creating manager programs quickly and easily.

The Agent

The agent is an application that runs on a system, which is the object of the management activities. The agent runs on the device which is to be managed: a host, gateway, terminal server, repeater, hub, switch, concentrator, bridge etc. Simply, the agent runs off each node on the network. It collects network and terminal information as specified in the MIB (Management Information Base). A common programming job is extending an agent to accomplish the specific needs of the network.

The MIB

The information that is made available to managers by an agent is organized in a hierarchical collection of objects, known as a management information base (MIB). The objects of the MIB are organized in a logical tree. Each object is indexed by its position within the tree. Frequently specific subtrees are also identified as MIBs themselves.

The MIB was defined by the Internet Engineering Task Force (IETF) for SNMP management. It is the repository for all information used to manage a network device. It describes the name, object identifier, data type, and accessibility of every data item that can be read or written via SNMP. Subtrees of the MIB are defined by IETF, vendors, or other organizations.

Structure

The MIB is structured like a tree. At the top of the tree is the most general information available about a network. Each branch of the tree then gets in more details into a specific network area, with the leaves of the tree as specific as the MIB can be. The top of a LAN MIB tree is usually referred to as the *Internet*.

The MIB uses **Abstract Syntax Notation One (ASN.1)** to name all variables. ASN.1 defines a hierarchical namespace, so the name of each variable reflects its position in the hierarchy. Each part of the hierarchy has been assigned a label, and the name is written as a sequence of labels that denote subhierarchies, with dots separating the labels. The label for the most significant hierarchy appears on the left. Thus, the MIB variable in the ip subhierarchy that counts incoming IP datagrams, *ipInReceives*, is named

internet. mgmt. mib-2. ip. ipInReceives

As the example shows, MIB names can be quite long. Of course, names for items in tables will be even longer than names for simple variables because they contain additional labels that encode the index of the table entry and the field requested in that entry.

When sending and receiving messages, SNMP does not store variable names text strings. Instead, it uses a **numeric form** of ASN.1 to represent each name. Because the numeric representation is more compact than a textual representation, it saves space in packets.

The numeric form of ASN.1 assigns a unique (usually small) integer to each label in a name, and represents the name as a sequence of integers. For example, the sequence of numeric labels for the name of variable *ipInReceives* is *1.2.1.4.3*.

The sequence of numbers seen above was created this way:

- 1 ... *internet* is the first (and at the top),
- 2 ... *mgmt* is the second child of *internet*,
- 1 ... *mib-2* is the first (and only) child of *mgmt*,
- 4 ... *ip* is the fourth child of *mib-2* and finally
- 3 ... reflects that *ipInReceives* is the third child of *ip*.

When they appear in an SNMP message, the numeric representation of simple variable names has a zero appended to specify that the name represents the only instance of variable in the MIB, so the exact form becomes: *1.2.1.4.3.0*

How SNMP works

SNMP (version 1) allows a manager to view information from an agent by sending a GET request or to modify information within an agent by sending a SET request. Frequently a request to view or modify such information is accompanied by a 'community name', which acts as a password, thus providing a basic level of security. Communication between a manager and agent is commonly via this request/response exchange. An agent can also provide unsolicited information to a manager by sending a TRAP.

SNMP allows a manager to also access values via a GET-NEXT request. Unlike GET or SET requests, the GET-NEXT request does not specify the name of the item to be retrieved. Instead, it specifies a name, and asks the server for the name and value of the variable that occurs next in the lexical sequence. This command is usable for tables of unknown size.

SNMP (version 2) comes with further improvements:

- it enables the exchange of information between managers
- it revamps security by replacing the community name with the context and other related concepts

SNMP itself is transported across the network in traditional packets using standard network protocols.

SNMP Traps

As we mentioned previously, the agent can provide unsolicited information to a manager. Although SNMP is normally used in a synchronous manner in which systems select each other periodically, it does include a mechanism for asynchronous events. In SNMP, these are referred to as **traps**.

Traps are sent by agents to management stations to signal the occurrence of an asynchronous event.

Trap Console is a piece of software, which allows the user to manage traps, respond to them, filter undesired traps, etc.

A trap holds information of a specific structure defined by [rfc1215](#) (see the section 'For Further Study' on page 79).

Because traps are asynchronous messages signalling an event that may require attention, there are no responses from the management station to them. The agent may generate seven kinds of trap:

- coldStart
- warmStart
- linkDown
- linkUp
- authenticationFailure
- egpNeighborLoss
- enterpriseSpecific
- A short description of them follows here:

coldStart

Implies that the sending agent is reinitializing itself with significant changes in its configuration, for example, after a reboot.

warmStart

Implies that the sending agent is reinitializing itself with no changes in its configuration.

linkDown

Implies a failure in one of the communication links of the sending agent.

linkUp

Implies that one of the communication links of the sending agent has been restored to the normal running status.

authenticationFailure

Implies that an instance of authentication failure has occurred at the sending agent.

egpNeighborLoss

Implies that an EGP peer relationship of the sending agent's EGP protocol has been lost.

enterpriseSpecific

Implies that an enterprise-specific event has occurred.

Implementation

SNMP is implemented as a client/server (using UDP - User Datagram Protocol). The agent acts as the server and the manager as the client.

The agent (server) machine runs a process called *snmpd*, which continually listens for get- and set-requests from a manager. The manager (client) machine typically runs a sophisticated graphical application, which uses get- and set-requests to perform tasks required by the human user.

For Further Study

This chapter is intended only as an introduction to SNMP. For further study, see the following several books and Internet resources:

Paul Simoneau	Hands-On SNMP <i>McGraw Hill Publishing, 0079130755</i>
Marshall Rose	Simple Book <i>Prentice Hall, 0134516591</i>
Dr. Sidnie Feit	SNMP <i>McGraw Hill Publishing, 0070203598</i>
Evan McGinnis	Understanding SNMP MIBs <i>Prentice Hall, 0134377087</i>
http://www.faqs.org/rfcs/rfc1215.html	RFC1215.

Contact Information

CSCare Inc. welcomes your comments and suggestions on the quality and usefulness of Trap Console. You can contact us on

trapconsole@cscare.com, or

CSCare Inc.
2880 Zanker Road, Suite 203
San Jose
CA 95134

Tool-free: 1-866-783-0663

Phone: (408) 806-6267

Fax: (408) 904-5620

For the latest news and information about Trap Console, please check the CSCare Web site at

<http://www.cscare.com/trapconsole>.

Appendix

Customizing trapconsole.cfg

When starting, Trap Console loads its settings from the configuration file `trapconsole.cfg`. This appendix lists all parameters in this file, which can be set according to your needs. After changing parameters, do not forget to restart Trap Console for this change to take effect.

- **HttpServerPort** – The HTTP port on which the Trap Console's Web server listens. To get more information, see the sections '[Starting Trap Console on a Port other than 6610](#)' on page 15 and '[Trap Console Manager](#)' on page 19.

Default: `HttpServerPort=6610`

- **HttpSessionTimeout** – The maximum period of inactivity for the Web browser, set in seconds.

Default: `HttpSessionTimeout=600`

- **HttpQueueLength** – The maximum amount of HTTP requests that Trap Console accepts for processing if all HTTP server threads are busy.

Default: `HttpQueueLength=256`

Note: As Trap Console is a single-user system, it is not necessary to modify this value.

- **HttpThreads** – The maximum allowed number of embedded HTTP server threads running concurrently. Each thread handles a single user's request.

Default: `HttpThreads=32`

Note: As Trap Console is a single-user system, it is not necessary to modify this value.

- **TrapQueueLength** – The maximum amount of trap packets that Trap Console is allowed to place into its packet queue, when trap-processing threads are busy. Increase this number in order to minimize the amount of dropped packets reported on the [Runtime/UDP Trap Receiver](#) page.

Default: `TrapQueueLength=256`

Note: You can also change this value on the [Runtime/UDP Trap Receiver](#) page (see the section '[Tuning Trap Console](#)' on page 57).

- **TrapQueues** – The amount of trap processing queues. Increasing this number improves the speed with which traps are being retrieved from the underlying UDP protocol stack, but it adds high-priority threads to Trap Console, which may negatively affect the overall system performance.

Default: `TrapQueues=5`

- **TrapThreads** – The maximum amount of trap processing threads Trap Console is allowed to run at the same time. Trap Console is concurrently processing incoming traps (additional ones wait in the trap queues for processing). Increase this value to increase Trap Console's throughput.

Default: `TrapThreads=32`

Note: You can also change this value on the [Runtime/UDP Trap Receiver](#) page (see the section '[Tuning Trap Console](#)' on page 57).

- **TrapPort** – The UDP port on which Trap Console listens for SNMP traps. By default SNMP traps are being sent to the well-known SNMP trap port 162. However, especially for testing purposes or for security reasons, it is possible to configure Trap Console to listen for SNMP traps on a different UDP port.

Default: `TrapPort=162`

- **SocketTimeout** – The socket communication timeout in milliseconds used in all actions processing the Telnet uplink communication.

Default: `SocketTimeout=120000`

- **TcpTrapPort** – The TCP trap port on which Trap Console listens for SNMP traps. By default SNMP traps are being sent to the SNMP trap port 162. However, especially for testing purposes or for security reasons, it is possible to configure Trap Console to listen for SNMP traps on a different TCP port.

Default: `TcpTrapPort=162`

- **TcpTrapInterface** – Local interface that TCP receiver listens on (default none, means any available).

Default: `TcpTrapInterface=`

- **TcpTrapSocketTimeout** – TCP receiver socket timeout in milliseconds.

Default: `TcpTrapSocketTimeout=20000`

- **TcpTrapHostNameLookup** – Name lookup for accepted connections (attempt peer IP address)

Default: `TcpTrapHostNameLookup=1`

- **TcpTrapQueueLimit** – TCP message queue size limit.

Default: `TcpTrapQueueLimit=256`

- **TcpTrapQueueThreadLimit** – TCP connection processing thread count limit.

Default: `TcpTrapQueueThreadLimit`

- **mibs.dir** – Path to the directory containing MIB files.

Default: `mibs.dir=mibs`

- **mibs.file** – Path to the `mibs.txt` file, containing the list of all registered MIBs. If it is required, use `\\` for backslash.

Default: `mibs.file=mibs.txt`

- **AppLogLimit** – The maximum size of the application log file in KB.

Default: `AppLogLimit=100`

- **AppLogLevel** – Specifies the logging level;

0 = no log messages, 1 = standard, 2 = detailed, and 3 = debug log messages.

Default: `AppLogLevel=1`

- **MaxLogFileSize** – Maximum size of action log files in kB (see the section 'Log Files' on page 54).

Default: `MaxLogFileSize=100`

- **LogActionAbsPaths** – Disables or enables absolute paths for action log files.

Default: `LogActionAbsPaths=1`

The following **SMTP server** settings can be configured here as well:

- **admin.address** – is an e-mail address of the Trap Console administrator. Trap Console will send notification messages to this address
- **sender.address** – is an e-mail address Trap Console uses as the sender of e-mails
- **smtp.server** – The IP address/name of the mailserver Trap Console uses to sent all e-mail messages out. More than one server to backup the SMTP server in case of failure of the first SMTP server can be specified.

Default: `smtp.server=127.0.0.1:25`

If more servers should be supported, separate them with semicolon:

`smtp.server=127.0.0.1; 10.0.0.10:25; smtp.someserver.com`

- **smtp.spool** – Relative or absolute path to the message spool directory.
Default: `smtp.spool=spool`
Note: When 'smtp.spool' path contains backslashes on MS Windows operating system, every backslash in the path must be doubled ('\'). Thus, for example if the spool directory is 'c:\sender\spool', in the configuration file it must be set as 'c:\\sender\\spool'.
- **smtp.timeout** – Defines the time period in milliseconds for sending e-mail messages out from the spool.
Default: `smtp.timeout=10000` (i.e. 10 seconds)
- **smtp.sotimeout** – Socket timeout in milliseconds, when communicating with SMTP server.
Default: `smtp.sotimeout=300000` (i.e. 5 minutes)
- **smtp.logging** – Specifies the logging level; 0 = no log messages, 1 = standard, 2 = detailed, and 3 = debug log messages.
Default: `smtp.logging=2`
- **smtp.limit** – The maximum size of the SMTP log file in KB.
Default: `smtp.limit=100`
- **smtp.threads** – The number of threads sending e-mails out from the spool directory.
Default: `default: smtp.threads=1`
- **SyslogUdpPort** – UDP port for Syslog receiver listening for syslog messages. Syslog receiver is initially disabled.
Default: `SyslogUdpPort=0`
- **SyslogUdpInterface** – Local interface for Syslog UDP receiver to bind (default none, means any available)
Default: `SyslogUdpInterface=`
- **SyslogUdpQueueLimit** – Maximum length of syslog message packet queue.
Default: `SyslogUdpQueueLimit=256`
- **SyslogUdpQueueThreadLimit** – Maximum number of syslog message packet processing threads.
Default: `SyslogUdpQueueThreadLimit=32`
- **SyslogTcpPort** – TCP port for Syslog receiver listening for peer syslog connections. Syslog receiver is initially disabled.
Default: `SyslogUdpPort=0`
- **SyslogTcpInterface** – Local interface for Syslog TCP receiver to bind (default none, means any available)
Default: `SyslogTcpInterface=`
- **SyslogTcpSocketTimeout** – TCP syslog receiver socket timeout in milliseconds.
Default: `SyslogTcpSocketTimeout=20000`
- **SyslogTcpQueueLimit** – Maximum length of syslog connection queue.
Default: `SyslogTcpQueueLimit=256`
- **SyslogTcpQueueThreadLimit** – Maximum number of syslog connection processing threads.
Default: `SyslogTcpQueueThreadLimit=32`

Command Line Parameters

This appendix provides a list of Trap Console command line parameters, which allow customizing Trap Console at its startup.

If Trap Console starts without any command line parameters (see the section 'Starting Trap Console' on page 13), it reads the configuration from the `trapconsole.cfg` file.

User can change the Trap Console configuration by adding command line parameters during startup. Trap Console command line parameters could be used with both '-' and '/' prefixes, and are case insensitive. They have higher priority than the ones saved in the `trapconsole.cfg` file.

The command line with a parameter looks as follows:

```
java -jar tc.jar [-parameter]
```

The following parameters can be used:

- **-d <path>** – sets different working directory
- **-daemon** – runs without the console thread. Useful for services (see also the section 'Starting Trap Console in Daemon Mode' on page 15).
- **-t <number>** – time correction [milliseconds]
- **-h** – print the list of all command line parameters
- **-p <port>** – starts the HTTP server on the specified port (see also the section 'Starting Trap Console on a Port other than 6610' on page 15).
- **-ssl** – specifies SSL operation for the HTTP server
- **-if <host or IP address>** – binds the local interface to the specified server. This is useful when there are more interfaces connected to the server and you wish to allow clients connecting to Trap Console console with their browsers via only one particular interface (specifying its host or IP address).
- **-stop [<timeout – seconds>]** – stops the running instance of the application, timeout defined in seconds

Example:

```
java -jar tc.jar -daemon
```

SNMP Command Line Utility

Trap Console provides console mode SNMP utility. It is an SNMP command generator utility. Its launch script is located in Trap Console installation directory.

On Microsoft Windows,

```
snmp.bat
```

on Linux,

```
sh snmp.sh
```

Format of command line parameters generally starts with SNMP command name followed by variable names or values if applicable.

Default application parameters can be modified by adding appropriate switches.

All command line parameters can be loaded from a prepared parameter file.

```
snmp (get|getnext|set) { variable [value] } [parameters]
```

```
snmp getbulk nonRepeaters maxRepetitions { variable } [parameters]
snmp trap|inform trapName { trapValue } [parameters]
snmp traps|informs [parameters] ... sends all traps to the destination
snmp walk mibObjectName [parameters]
snmp browse mibObjectName [parameters]
snmp table tableName { column } [parameters]
snmp bulktable tableName { column } [parameters]
snmp decode [dump file name] .... decodes the dump file (def. dump.log)
snmp compile .... recompiles all mibs
snmp ver ... prints version information
snmp (help|-?|-h) ... prints usage information
snmp @fileName ... fileName contains parameters
```

Parameters:

- **-a <address or host name> [: <port>]** ... default = 127.0.0.1
- **-tcp** ... use SNMP over TCP transport
- **-c <community>** ... default = public
- **-w <timeout ms>** ... default = 3000ms
- **-r <retries>** ... count of retries (default = 0)
- **-t** ... repeat until interrupted
- **-n <count>**
- **-d <delay ms>** ... delay between actions
- **-m <mib name>** ... use a MIB [default is RFC1213-MIB]
- **-v2** ... use SNMP v2 PDUs
- **-ts <nonnegative long value>** ... timestamp value for trap pdu
- **-ag <agent address or host name>** ... agent address for trap pdu
- **-lif <local if address>** ... local if to send packets through
- **-nr** ... non-repeaters for bulk operations [default is 0]
- **-mr** ... max-repetitions for bulk operations [default is 2]
- **-od** ... print time into output
- **-ov** ... print variable name into output
- **-oi** ... print variable OBJECT IDENTIFIER into output
- **-ox** ... print variable syntax into output
- **-oa** ... print agent address and name
- **-b** ... print values without any formatting, octet strings as string of hex values
- **-s** ... do not print statistics
- **-e** ... wait for enter at the end
- **-log** ... log all packets to dump.log file
- **-dir <directory>** ... path to the working directory

Macros

Trap actions macros. See [Actions](#) section for more information.

General Macros

- **%gate%** - identification of the Trap Console application
- **%gateVersion%** - version of the Trap Console application
- **%gateAddress%** - IP address of the host, where Trap Console runs
- **%gateName%** - host name of the host where Trap Console runs
- **%timeticks%** - current time in milliseconds (when performing an action)
- **%time%** - current time (when performing an action)
- **%shortTime%** - current time in short form – HOUR:MINUTE:SECOND
- **%shortDate%** - current date in short form - YEAR.MONTH.DAY

Trap Message Macros

- **%protocol%** - "UDP" or "TCP" string depending on the protocol used to carry the trap message
- **%senderAddress%** - IP address of the SNMP agent from which a trap actually arrives
- **%senderName%** - host name of the SNMP agent from which a trap actually arrives. If not available, IP address displays. If the trap is forwarded, the sender address/name changes to IP address/name of the trap.
- **%snmpVersion%** - inbound SNMP message version. Possible values are: "0" for SNMPv1, "1" for SNMPv2c and "3" for SNMPv3.
- **%community%** - SNMP community string received in the inbound SNMP trap
- **%encoding%** - SNMP v2 PDU encoding. Can be "trap" or "inform-request"

Trap PDU Macros

- **%enterpriseld%** - SNMP enterprise (e.g. trap group) of the inbound SNMP trap in the raw form (e.g. 1.3.6.1....)
- **%agentAddress%** - IP address of the SNMP agent which generated the trap
- **%agentName%** - host name of the SNMP agent, which generated the trap. If not available, IP address displays.

Note that SNMP v2 does not support agent address. SNMP v2 traps always use the sender address. It means macros %agentAddress/Name% and %senderAddress/Name% expand to the sender address/name.

- **%genericTrap%** - trap value for SNMP traps as defined for SNMP v1 trap PDU
- **%specificTrap%** - trap value for enterprise specific traps as defined for SNMP v1 trap PDU
- **%timeStamp%** - as indicated by the SNMP agent sending the trap.
- **%trapOID%** - OID (Object Identifier) of the trap being processed.
- **%varOIDN%** - OID of the n-th variable in the optional variable list of the trap (sample: var1, var2, ...). Starts from 1.
- **%varOIDs%** - comma-separated list of OIDs of all variables received in the variable list of the trap

- **%varOIDvalN%** - variable OID and value of the n-th variable-value pair received in the trap. Formatted as "variableOID(value)" (sample: sysDescr(HP Hub), ifIndex(1)).
- **%varOIDvals%** - all the variable OID-value pairs in the comma separated string
- **%valN%** - n-th value in the optional variable list of the trap (sample: val1, val2, ...). Starts from 1.
- **%vals%** - comma-separated list of values of all variables received in the variable list of the trap
- **%valse%** - the same as %vals%, values are enclosed in apostrophes
- **%valsee%** - the same as %valse%, values are enclosed in two apostrophes, in addition, every apostrophe in the value content is converted to four apostrophes. Used for some procedures. This macro works properly only if used in actions derived from the 'Write trap to the database' type action.
- **%<OID>%** - value of the variable given OID, e.g.%1.3.6.1.4.1.5980.102.2.1%

Trap MIB Macros

- **%mib%** - indicates whether the inbound trap was resolved using registered MIB files. Can be "resolved" or "unresolved".
- **%name%** or **%trapName%** - name of the trap being processed.
- **%status%** - status of the trap being processed.
- **%description%** or **%trapDescr%** - description of the trap being processed.
- **%reference%** - reference of the trap being processed.
- **%enterprise%** - SNMP enterprise name (e.g. trap group) of the inbound SNMP trap, translated to the string using registered MIB files.
- **%trapValue%** - the #TRAP-TYPE clause value
- **%hashType%** - the #TYPE clause value for the trap being processed.
- **%hashSummary%** - the #SUMMARY clause value for the trap being processed.
- **%hashArguments%** - the #ARGUMENTS clause value for the trap being processed.
- **%summary%** - the #SUMMARY clause value with #ARGUMENTS substituted with processed trap variables.
- **%hashSeverity%** or **%severityName%** - the #SEVERITY clause value for the trap being processed.
- **%severity%** - severity number. The #SEVERITY clause value converted to a respective number.
- **%hashTimeIndex%** - the #TIMEINDEX clause value for the trap being processed.
- **%hashHelp%** - the #HELP clause value for the trap being processed.
- **%hashHelpTag%** - the #HELPTAG clause value for the trap being processed.
- **%hashState%** - the #STATE clause value for the trap being processed.
- **%hashGeneric%** - the #GENERIC clause value for the trap being processed.
- **%hashCategory%** - the #CATEGORY clause value for the trap being processed.
- **%hashSourceID%** - the #SOURCE_ID clause value for the trap being processed.
- **%varDescrN%** - description of the n-th variable in the optional variable list of the trap (sample: var1, var2,...). Starts from 1.
- **%varN%** - name of the n-th variable in the optional variable list of the trap (sample: var1, var2,...). Starts from 1.
- **%vars%** - comma-separated list of names of all variables received in the variable list of the trap.

- **%varvalN%** - variable name and value of the n-th variable-value pair received in the trap. Formatted as "variable(value)" (sample: sysDescr(HP Hub), ifIndex(1)).
- **%varvals%** - all the variable-value pairs in one comma-separated string.
- **%<name>%** - value of the variable with the given name, e.g. %nodeName%

Additional Date and Time Macros for Current Time

- **%numberDay%** - 01 up to 31
- **%shortDay%** - Mon, Tue, Wed, Thu, Fri, Sat, Sun
- **%day%** - Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
- **%numberMonth%** - 01 up to 12
- **%shortMonth%** - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
- **%month%** - January, February, March, April, May, June, July, August, September, October, November, December
- **%shortYear%** - year in short form (01, 02 ...)
- **%year%** - year in long form (2001, 2002 ...)
- **%hour%** - current hour in 12 hour format
- **%hour24%** - current hour in 24 hour format
- **%minute%** - current minute
- **%second%** - current second
- **%amPm%** - AM or PM

Expressions

Expression on trap parameters and variables has to be a logical expression evaluating to a boolean value. Expressions can be defined (recursively) as follows:

- The name of a general trap parameter or trap specific variable is an expression.
See the list of available trap parameter names [below](#).
The name containing spaces can be enclosed in single quotes.
- Constants are expressions.
Integer constant is a sequence of digits (0..9)
String constants are enclosed in double quotes.
Boolean constant is one of the keywords: "true" and "false".
Examples of constants: "string constant", "xyz", 12345, true, false
- Unary expression: unary operator (+, -, !, ~) followed by an expression.
Examples: -1, !(a == 7)
- Binary expression: expression, binary operator and second expression.
Where binary operator is one of +, -, *, /, %, <, <<, >, >>=, ==, !=, &, |, ^, &&, ||.
Examples: a + b, a - b, a < b, a || b
- Conditional expression: noted as <expr1> ? <expr2> : <expr2> Example: a < b ? 1 : 2
- Function expression: the function name followed by a comma separated argument list enclosed in parentheses.
Examples: contains(senderName,"acme"), substring(senderName,10)

During the process of evaluation of an expression it is assigned a value.

The value of atomic expressions, e.g. variables and constants, is the value of the constant or the variable.

The binary operations +, -, *, /, % have their usual meaning (addition, subtraction, multiplication, division and remainder) and both of their operands must be integer values. One exception is +, which has a meaning of concatenation of strings if at least one of its operands is a string value. In that case, the other operand is converted to the string value as well.

The binary operations <<, >>, >>> denote bit shifts (left shift, right shift and unsigned right shift) and both of their operands must be integer values.

The binary operations ==, !=, <, <=, >, >= denote operations used for comparison of values (their meaning is equal, not equal, less than, less or equal, greater than, greater or equal). Both operands of these operations must be of the same type. In the integer values comparison, numerical values are compared. In the string values comparison, strings are compared lexicographically. In case of boolean values, only operations ==, != are possible.

The operands of operations &, |, ^ must be both of boolean or both of integer type. In case of boolean values, these operations denote 'logical and', 'logical or' and 'logical exclusive or'. In case of integer values, these operations are performed on every bit of the values.

The operands of operations && and || must be boolean values.

The operand of the unary operations +, - and ~ must be an integer value. The meaning of these operations is: unary plus, unary minus (negation) and bit complement. The operand of the unary operation ! must be a boolean value and the meaning of this operation is logical negation.

The ternary operation ? : is evaluated as follows: Evaluation starts with the first operand. It must be a boolean value. If it has the value **true**, only the second operand is evaluated then, and the result is the value of this operand. If the first operand has the value **false**, only the third operand is evaluated, and the result of the whole operation is the value of the third operand. The type of values to which the second and the third operand would evaluate should be the same.

Functions

The following predefined functions can be used in expressions:

- Function **length** expects one argument of string type. It returns the length of its argument - the number of characters in the string. The result is an integer value.
- Function **equals** has two arguments. Both arguments must be of string type. The function returns a boolean value, which is true if the first argument equals the second argument, and false otherwise.
- Function **equalsIgnoreCase** is the case insensitive version of the **equals** function
- Function **contains** has two arguments. Both arguments must be of string type. The function returns a boolean value which is true if the first argument contains the second argument as a substring, and false otherwise.
- Function **containsIgnoreCase** is the case insensitive version of the **contains** function
- Function **indexOf** can have two or three arguments. The first and second argument must be of string type. The third argument (if it is present must be of integer type). The function returns an index of the first occurrence of the second argument in the first argument, or value -1 if the second argument is not found in the first argument. The index of the first character in the string is 0. If the third argument is present, search is started from the position given by the third argument.
- Function **indexOfIgnoreCase** is the case insensitive version of the **indexOf** function
- Function **lastIndexOf** can have two or three arguments and its meaning is exactly the same as that of the function **indexOf** with the exception that the last and not the first occurrence of the second argument is found in the first argument.
- Function **lastIndexOfIgnoreCase** is the case insensitive version of the **lastIndexOf** function.
- Function **substring** can have two or three arguments. The first argument must be of string type, the following arguments must be of integer type. If the function has three arguments, it returns the substring located in the value string of the first argument starting from the character with an index given by the second argument to the character with an index given by the third argument. If the function has two arguments, the result is the same as if the third argument had the same value as the length of the first argument. An error occurs if the second argument is negative or greater than the third argument (or greater than the length of the string), or if the third argument is greater than the length of the string.
- Function **startsWith** has two arguments. Both arguments must be of string type. The function returns a boolean value, which is true if the first argument starts with the second argument, and false otherwise.
- Function **startsWithIgnoreCase** is the case insensitive version of the **startsWith** function.
- Function **endsWith** has two arguments. Both arguments must be of string type. The function returns a boolean value, which is true if the first argument ends with the second argument, and false otherwise.
- Function **endsWithIgnoreCase** is the case insensitive version of the **endsWith** function.
- Function **containsVariable** expects one argument of string type. It returns true if the trap contains a variable of the given name, false otherwise.

Priority of Operations

The order of evaluation of an expression is given by priorities of operations. The default priority can be changed using parentheses - the expression in the parentheses is always evaluated first. Generally, unary expressions have higher priority than binary expressions, and binary expressions have higher priority than the ternary expression ?:

The only ambiguity can appear in binary operations, because they use infix notations. The following order of priorities of binary operations (the highest priority at the top) is defined:

1. *, /, %
2. +, -
3. <<, >>, >>>
4. <, <=, >, >=
5. ==, !=
6. &
7. ^
8. |
9. &&
10. ||

Example:

```
a + b * c
```

has the same meaning as

```
a + ( b * c )
```

The operations with the same priority are associated to the left, so

```
a + b + c
```

has the same meaning as

```
( a + b ) + c
```

Variables

Since Trap Console 1.5 release version, expression variable names and macro names are unified. See [Macros](#)

description for available names. Variable name will be the same as macro, without “%” delimiters. Trap specific variables can be used in expressions via their names defined in appropriate MIBs.

Examples:

Expressions if evaluate to true can activate a set action. The following examples presume that the used traps are allowed for the rule.

- `contains(alarmName, "fan")`

This expression evaluates to true if the following condition is fulfilled: the value of the 'alarmName' variable ('csCareTrap' trap) contains a substring: fan. The 'substring' function is case sensitive.

- `contains(alarmName, "'fan'")`

Let's assume that in the 'alarmName' variable there is a string: `Server 'fan' 2`. To match the substring ('fan') including the single quotes, the whole searched substring must be placed in double quotes in the 'contains' function's second argument: `"'fan'"`.

- `contains(alarmName, "\"fan\"")`

An example of searching for a substring with double quotes. This example evaluates to true if the 'alarmName' variable contains the following substring including double quotes: `"fan"`.

- `substring(alarmName, 4, 7) == "off"`

If there is a need to search for a substring at an exact position in the value string (of the 'alarmName' variable in this case), use the 'substring' function. The expression above evaluates to true if the substring which starts with a character at position '4' and ends with the 7-th character in

the 'alarmName' variable matches the 'off' substring.

Note: The position of the first character in a string is indexed as '0'. If we have an expression: `substring(variable, 4, 7)`, and the variable has a value: 0123456789, then the substring above returns: 456.

- `startsWith(trapName, "link") && ifIndex == 3`

This expression evaluates to true only if the incoming trap's name starts with the 'link' substring (linkUp, linkDown, ...) and in the same time the trap's 'ifIndex' variable has a value that equals to 3.

- `mib && length(trapDescr) > 30`

This expression evaluates to true if a trap is resolved (it has a MIB) and the length of the trap description (i.e. the number of characters in 'trapDescr' variable value) is greater than 30.

- `trapName == "egpNeighborLoss" && substring(egpNeighAddr, 0, 7) == "10.0.0."`

This expression evaluates to true only if the incoming trap is named 'egpNeighborLoss', and in the same time, the EGP neighbor IP address substring is exactly '10.0.0.' and starts with the first character (argument '0') and ends with the 7-th character (argument '7') in the 'egpNeighAddr' string.

- `containsVariable("nodeName") && indexOf(nodeName, "PROXY") == -1
or
containsVariable("nodeName") && !contains(nodeName, "PROXY")`

The both expressions are equal and evaluate to true if an incoming trap contains a 'nodeName' variable and in the same time the 'nodeName' variable doesn't contain the 'PROXY' substring.

- `equals(trapName, "serviceFailure") && equalsIgnoreCase(cswServiceName, "FTP") && endsWithIgnoreCase(cswServiceHost, "acme.com")`

Presumably, the csWatchTrap enterprise is allowed for this rule. The above-mentioned expression evaluates to true if an incoming trap is named 'serviceFailure', is sent from an ftp server, and the host's domain address ends with the 'acme.com' string. The case of the 'ftp' string and the domain address string is ignored.

- `equalsIgnoreCase(enterprise, "snmp") || trapName == "csCareTrap" || equals(trapName, "serviceFailure")`

This expression allows finer selection of traps than a rule filter can do. The equality operator '==' performs the same way as the 'equals' function, but it cannot ignore the case (in contrary with 'equalsIgnoreCase').

Index

- A**
- Application log file 7, 16, 58, 73
- C**
- Command Line Parameters 14, 19, 84
- D**
- Data Source Name..... 64, 66, 67
- E**
- e-mail address 10, 45, 59
- F**
- File
- ACTION_*.cfg..... 34, 38, 53
 - readme.txt..... 19, 73
 - RULE_*.cfg..... 29, 53
 - server.log..... 58
 - smtpsender.log 53, 59, 82
 - trapconsole.bat..... 70
 - trapconsole.cfg..... 19, 53, 59, 80, 84
 - traps.log..... 30, 40, 53
 - unresolved.log..... 28, 30, 40, 53
- Folder
- TrapConsole.. 2, 8, 12, 16, 19, 22, 25, 29, 34, 38, 39, 74, 79
 - TrapConsole\mibs 22, 23, 25
 - TrapConsole\msgspool..... 59, 82
 - TrapConsole\Templates 19
- I**
- Installation program 8
- IP address...7, 10, 19, 26, 27, 33, 45, 47, 49, 52, 58, 66, 68, 71, 73
- J**
- Java Secure Socket Extension..... 17
- Java Virtual Machine 11, 43, 64, 71, 72
- JDBC.....6, 42, 43, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72
- JDBC-ODBC Bridge..... 64, 65, 67, 68
- JSSE 17
- K**
- Keystore 17
- Keytool..... 17
- L**
- License
- Key..... 7, 21, 22
 - Overflow 7, 10, 58, 59, 73
- Log file..... 28, 30, 40, 41, 42, 53, 58, 59, 60, 81
- M**
- Macros..... 6, 38, 52
- MIB database5, 6, 20, 22, 23, 24, 25, 28, 30, 51, 75, 76, 81
- Microsoft Access Database 64
- Microsoft Windows 98/95..... 8, 12, 13
- Microsoft Windows NT 8, 12, 13, 68, 71, 72
- N**
- NT service 15, 16, 65, 68
- O**
- Oracle Database 64, 67, 69
- P**
- Password 19, 20, 21, 44, 68, 70
- S**
- SMTP server 10, 58
- SNMP
- Agents 7, 66, 68, 71, 73, 76, 78
 - Community..... 28, 52
 - Trap.....2, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 46, 47, 53, 57, 66, 73, 75, 77, 79, 80, 81, 84
 - Versions 6, 24, 28, 48, 52, 77
- SSL..... 17, 18
- SSL keystore 17
- T**
- TCP port..... 15
- TCP/IP..... 71
- Threads..... 57, 58, 80
- Trap
- Description 26
 - Enterprise 25, 26, 27, 31, 51, 78
 - Variables 28, 52
- Trap Console
- Actions...5, 10, 19, 20, 26, 28, 30, 34, 38, 39, 40, 41, 42, 44, 45, 46, 48, 49, 50, 51, 52, 53, 58, 65, 67, 69, 70, 72, 80
 - Administrator 5, 10, 59, 64, 65, 66, 67, 68
 - Evaluation 7
 - Login 19, 20, 21, 22
 - Logout 21, 22
 - Manager 19, 20, 21, 22, 29
 - Rules...5, 20, 25, 26, 27, 28, 29, 30, 31, 33, 34, 40, 53
- Troubleshooting 12, 15, 18
- U**
- UDP port 15, 27, 45, 47, 52, 78, 80, 81
- UNIX..... 11, 19
- Unresolved traps..... 6, 28, 30, 40, 53
- User Interface..... *See* Web-browser
- W**
- Web server 15, 17, 19, 80, 84
- Web-browser..... 5, 7, 19, 21, 22, 73

Z

ZIP file 8