

English



FUJITSU Software BS2000

# SNMP Management (NET-SNMP) for BS2000

User Guide

Valid for:

NET-SNMP V5.7

SNMP-AGENTS V1.0

Edition January 2019

## **Comments... Suggestions... Corrections...**

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

[bs2000services@ts.fujitsu.com](mailto:bs2000services@ts.fujitsu.com)

## **Documentation creation according to DIN EN ISO 9001:2015**

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

cognitas. Gesellschaft für Technik-Dokumentation mbH

[www.cognitas.de](http://www.cognitas.de)

## **Copyright and Trademarks**

Copyright © 2019 Fujitsu Technology Solutions GmbH.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

---

# Contents

<b>1</b>	<b>Preface . . . . .</b>	<b>7</b>
<b>1.1</b>	<b>Contents of the manual . . . . .</b>	<b>7</b>
<b>1.2</b>	<b>Target group . . . . .</b>	<b>7</b>
<b>1.3</b>	<b>Summary of contents . . . . .</b>	<b>8</b>
<b>1.4</b>	<b>Notational conventions . . . . .</b>	<b>9</b>
<b>1.5</b>	<b>Changes compared to the previous version . . . . .</b>	<b>10</b>
<b>1.6</b>	<b>README file . . . . .</b>	<b>11</b>
<b>2</b>	<b>Overview . . . . .</b>	<b>13</b>
<b>2.1</b>	<b>Basic features of the SNMP management architecture . . . . .</b>	<b>14</b>
<b>2.2</b>	<b>SNMP management in BS2000 - embedding and functionality . . . . .</b>	<b>17</b>
2.2.1	Product structure . . . . .	19
2.2.2	Structure of the SNMP collection in BS2000 . . . . .	22
2.2.2.1	SNMP daemon (snmpd) . . . . .	22
2.2.2.2	Agents . . . . .	23
2.2.3	User interface for the SNMP management of BS2000 . . . . .	23
2.2.4	Parallel operation of SNMP V6.x and NET-SNMP . . . . .	23
<b>2.3</b>	<b>Security considerations when using SNMP . . . . .</b>	<b>24</b>
2.3.1	Recommendations for general network and system security . . . . .	24
2.3.2	Recommendations for using the SNMP service safely . . . . .	24
2.3.2.1	Community strings for receiving SNMP requests . . . . .	25
2.3.2.2	Advanced security options for receiving SNMP requests . . . . .	26
2.3.2.3	Community strings and controlling access to MIB objects . . . . .	26
2.3.2.4	Community strings and sender addresses . . . . .	27
2.3.2.5	Recipient's addresses for SNMP traps . . . . .	27
2.3.2.6	Community string for SNMP traps . . . . .	27

<b>3</b>	<b>Installation and configuration</b>	<b>29</b>
<b>3.1</b>	<b>Software requirements</b>	<b>29</b>
<b>3.2</b>	<b>Installation in BS2000 OSD/BC</b>	<b>31</b>
3.2.1	Installation defaults	31
3.2.2	Delivery scope of NET-SNMP	33
3.2.3	Installing products manually	33
3.2.4	Uninstallation	34
<b>3.3</b>	<b>SNMP configuration in BS2000</b>	<b>35</b>
3.3.1	Listening addresses in BS2000	36
3.3.2	SNMP general configuration (snmp.conf)	37
3.3.2.1	Client behavior	37
3.3.2.2	SNMPv3 settings	39
3.3.2.3	Server behavior	40
3.3.2.4	MIB handling	41
3.3.2.5	Output configuration	41
3.3.3	AgentX configuration (agentx.conf)	44
3.3.4	Command line arguments	45
<b>3.4</b>	<b>Configuring NET-SNMP</b>	<b>47</b>
3.4.1	Configuring SNMP daemon snmpd (snmpd.conf)	47
3.4.1.1	Agent behavior	47
3.4.1.2	AgentX options	48
3.4.1.3	SNMPv3 configuration	49
3.4.1.4	SNMPv3 authentication	50
3.4.1.5	Access control	50
3.4.1.6	System Group	54
3.4.1.7	Active monitoring	55
3.4.2	DisMan Event MIB	56
3.4.3	DisMan Schedule MIB	59
3.4.4	Arbitrary Extension Commands	60
3.4.5	Configuration example	62
3.4.6	Reconfiguring the daemon	63
3.4.7	Configuring SNMP trap daemon snmptapd (snmptapd.conf)	64
3.4.7.1	snmptapd behavior	64
3.4.7.2	Access Control	64
3.4.7.3	Notification Processing	65
3.4.7.4	Logging	67
3.4.7.5	Format Specifications	68
<b>3.5</b>	<b>SNMP-AGENTS configuration</b>	<b>70</b>
3.5.1	Application Monitor agent configuration	70
3.5.1.1	Statements for the configuration file	71

3.5.1.2	Change in the configuration file during the current session . . . . .	81
3.5.2	Configuring the Console Monitor agent . . . . .	82
3.5.2.1	Positive message filter . . . . .	83
3.5.2.2	Structure of the positive message filter . . . . .	84
3.5.2.3	Negative message filter . . . . .	87
3.5.2.4	Modifying the configuration file during operation . . . . .	88
3.5.3	Configuring the Storage agent . . . . .	89
3.5.4	Configuring the openUTM agent . . . . .	93
3.5.4.1	Preparation . . . . .	93
3.5.4.2	Configuring the openUTM agent for monitoring several UTM applications . . . . .	94
3.5.4.3	Runtime environment . . . . .	96
3.5.4.4	Diagnostic documents . . . . .	97
3.5.5	Configuring the openSM2 agent . . . . .	99
3.5.6	Configuring the HSMS agent . . . . .	99
3.5.7	TCP-IP-AP configuration . . . . .	100
<b>4</b>	<b>Operations . . . . .</b>	<b>101</b>
<b>4.1</b>	<b>rc scripts . . . . .</b>	<b>102</b>
<b>4.2</b>	<b>NET-SNMP daemons and SNMP tools . . . . .</b>	<b>103</b>
4.2.1	SNMP daemon snmpd . . . . .	103
4.2.2	SNMP trap daemon snmptrapd . . . . .	104
4.2.3	SNMP tools snmpwalk, snmpget and snmpset . . . . .	105
<b>4.3</b>	<b>Starting and stopping the agents of SNMP-AGENT . . . . .</b>	<b>107</b>
4.3.1	Agents-specific options for starting agents manually . . . . .	108
<b>4.4</b>	<b>Starting BCAM, FTP and SESAM/SQL agents manually . . . . .</b>	<b>112</b>
<b>5</b>	<b>NET-SNMP functions . . . . .</b>	<b>113</b>
<b>5.1</b>	<b>Support of MIB-II (RFC 1213) . . . . .</b>	<b>113</b>
<b>5.2</b>	<b>Other MIBs supported by NET-SNMP . . . . .</b>	<b>114</b>
<b>5.3</b>	<b>Functionality of the Event services . . . . .</b>	<b>115</b>
<b>5.4</b>	<b>Functionality of the Scheduling services . . . . .</b>	<b>117</b>
<b>6</b>	<b>SNMP-AGENTS functions . . . . .</b>	<b>119</b>
<b>6.1</b>	<b>Application Monitor agent . . . . .</b>	<b>120</b>

# Contents

---

<b>6.2</b>	<b>Console Monitor agent</b> . . . . .	<b>121</b>
6.2.1	Acquiring console messages . . . . .	121
<b>6.3</b>	<b>Host Resources agent</b> . . . . .	<b>122</b>
<b>6.4</b>	<b>HSMS agent</b> . . . . .	<b>122</b>
<b>6.5</b>	<b>openFT agent</b> . . . . .	<b>123</b>
<b>6.6</b>	<b>openSM2 agent</b> . . . . .	<b>123</b>
<b>6.7</b>	<b>openUTM agent</b> . . . . .	<b>124</b>
<b>6.8</b>	<b>Spool agent</b> . . . . .	<b>124</b>
<b>6.9</b>	<b>Storage agent</b> . . . . .	<b>124</b>
<b>7</b>	<b>BCAM, SESAM/SQL and FTP agents functions</b> . . . . .	<b>125</b>
<b>7.1</b>	<b>BCAM agent</b> . . . . .	<b>125</b>
<b>7.2</b>	<b>FTP agent (from TCP-IP-AP package)</b> . . . . .	<b>126</b>
<b>7.3</b>	<b>SESAM/SQL agent</b> . . . . .	<b>126</b>
<b>8</b>	<b>Example for operating the management station</b> . . . . .	<b>127</b>
<b>9</b>	<b>Appendix: Procedure in the event of errors</b> . . . . .	<b>133</b>
<b>9.1</b>	<b>Format of the logging entries</b> . . . . .	<b>133</b>
<b>9.2</b>	<b>Configuring logging files of agents</b> . . . . .	<b>134</b>
<b>9.3</b>	<b>Debug options</b> . . . . .	<b>134</b>
	<b>Glossary</b> . . . . .	<b>137</b>
	<b>Related publications</b> . . . . .	<b>143</b>
	<b>Index</b> . . . . .	<b>147</b>

---

# 1 Preface

The delivery unit NET-SNMP V5.7 of the BS2000 operating system and the product SNMP-AGENTS V1.0 provide the basic functionality for linking BS2000 systems into SNMP-based management environments. NET-SNMP V5.7 and SNMP-AGENTS V1.0 provide network, system and application management capability via SNMP.

These components are supplemented by the product-specific agents BCAM, FTP (as part of interNet Services, delivered via TCP-IP-AP) and SESAM/SQL.

## 1.1 Contents of the manual

This manual describes the embedding of NET-SNMP and SNMP-AGENTS as well as the BCAM, FTP and SESAM/SQL agents into BS2000 systems, the installation and configuration steps required for operation, and operation itself. The agents required for monitoring, together with their MIBs, are described in detail.

In addition, detailed information about secure operation of SNMP management is described.

## 1.2 Target group

This manual is aimed at network operators and system administrators who integrate the BS2000 systems into an SNMP-based network, system and application management, or those who wish to operate such a system. Prior knowledge of the BS2000 operating system and the basic TCP/IP terms is assumed.

## 1.3 Summary of contents

This manual is structured as follows:

- Chapter 2: Overview

This chapter leads into the SNMP architecture, introduces fundamentals and describes the embedding into BS2000.

- Chapter 3: Installation and configuration

The installation requirements and installation itself are described in this chapter. The configuration steps on the BS2000 system are also shown in detail.

- Chapter 4: Operation

Chapter 4 describes the POSIX scripts for automatic startup and shutdown of the SNMP daemon and the agents, as well as the possibilities of starting daemons and agents in a manual way.

- Chapter 5: NET-SNMP functions

This chapter describes the groups in MIB-II (RFC 1213) and additional groups, which can be managed via the SNMP daemon. It also describes the implementation of DISMAN Monitoring (referred as the Event Services) and DISMAN Scheduling, which are also included in NET-SNMP as part of the SNMP daemon.

- Chapter 6: SNMP-AGENTS functions

This chapter describes the functionality and operation of the agents in SNMP-AGENTS including an overview of the respective MIBs.

- Chapter 7: BCAM, FTP and SESAM/SQL agents.

This chapter describes the functionality and operation of the BCAM and interNet Services agents.

Note: The SESAM/SQL agent is described in the SESAM/SQL documentation.

- Chapter 8: Example for operating the management station

- Appendix

The Appendix contains details on the procedures to be taken in the event of error.



## 1.4 Notational conventions

This manual uses the following symbols and formatting to emphasize particularly important sections of text:



for general information



for warnings

*Italics*

for file names, names of management windows and parameters, menu titles and menu items, as well as commands and variables included in continuous text.

*<angled brackets>*

designate variables, which have to be replaced by current values.

`fixed-width text`

for the representation of system inputs and outputs and file names in examples.

**command**

In the syntax description of commands, those parts that must be input unchanged (names of commands and parameters) are shown bold.

## 1.5 Changes compared to the previous version

SNMP has been re-implemented in BS2000 with several components: NET-SNMP, SNMP-AGENTS and the product-specific agents of BCAM, FTP and SESAM/SQL. These components replace the EMANATE-based products SBA-BS2, SSC-BS2, SSA-OUTM-BS2 and SSA-SM2-BS2.

In the following text, the previous implementation of SNMP in BS2000 will be referred to as "SNMP V6.x", the new solution will be called "NET-SNMP/SNMP-AGENTS".

This has led to the following changes in comparison to the SNMP Management V6.0A User Guide:

### **New product structure and new core software**

SNMP components:

- NET-SNMP V5.7 is a component of the BS2000 OSD/BC operating system. NET-SNMP V5.7 is based on the open source software Net-SNMP (BSD license).  
It contains the parts of MIB-II including the groups that are needed for openNet Server and interNet Services (IP, TCP, UDP). It also contains the functionality of both Event and Scheduling services, implemented by the Event and Scheduler agents of the previous SNMP versions V6.x.
- SNMP-AGENTS is an optional product, which contains the following agents:
  - Application Monitor
  - Console Monitor
  - Host Resources
  - HSMS
  - openFT
  - openSM2
  - openUTM
  - Spool & Print Services
  - Storage
- The product-specific agents of BCAM (private BCAM MIB), FTP and SESAM/SQL have also been re-implemented.

## Agents no longer included

The following agents are no longer offered:

- Supervisor Agent
- AVAS
- HIPLEX-AF
- OMNIS
- HTML Agent

## Other changes

- Manual start of the agents is now only possible using POSIX commands. Automatic start/stop scripts are still available, so it is possible to have an automatic start and stop of the agents on the POSIX startup/shutdown.
- The abbreviation in the names of the MIBs has been changed from "sni" to "fj".

## 1.6 README file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at <http://manuals.ts.fujitsu.com>. You will also find the Readme files on the Softbook DVD.

### *Information under BS2000*

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

```
SYSRME.<product>.<version>.<lang>
```

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

### *Additional product information*

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. This Release Notice is available online at <http://manuals.ts.fujitsu.com>.



---

## 2 Overview

SNMP stands for **S**imple **N**etwork **M**anagement **P**rotocol and was developed as a protocol for network management services in TCP/IP networks. The original task of SNMP was only the monitoring and administration of LAN components such as bridges, routers, hubs, etc. in heterogeneous networks with TCP/IP protocols. In the meantime, the application range of SNMP has been extended to include system and application management. SNMP does not stand for the protocol alone, but rather for the complete corresponding management system, in the same way that the term TCP/IP designates the complete network rather than just the protocol as such.

## 2.1 Basic features of the SNMP management architecture

The management platform is the central component of an SNMP installation. The management platform provides a well-structured display of the components it manages and offers easy operation. The network and all of its systems and applications can be monitored and controlled from the management platform. SNMP is not fixed to a particular platform.

The SNMP manager, also referred to as management station, resides in the management platform. The SNMP manager is an application that communicates with partner applications, the SNMP agents via SNMP over a TCP/IP network. Each managed component has an agent that provides the SNMP manager with current information about the component. The initiative for controlling the activities mainly rests with the SNMP manager, which ensures that the components to be managed only have to cope with a small volume of management tasks.

The basis for managing the components concerned is an exact description of the parts of these components that are to be administered (objects) in the MIB (Management Information Base). The MIB is the informational backbone of each Management Agent. It contains information on the characteristics, such as name, syntax, access rights and state, of each separate component. Specific MIBs are supplied by many hardware and software component manufacturers. The MIB coding is carried out in ASN.1 (Abstract Syntax Notation One). ASN.1 has been ratified by the ISO as a standard for the presentation layer (see ISO/IEC 8824 and 8825).

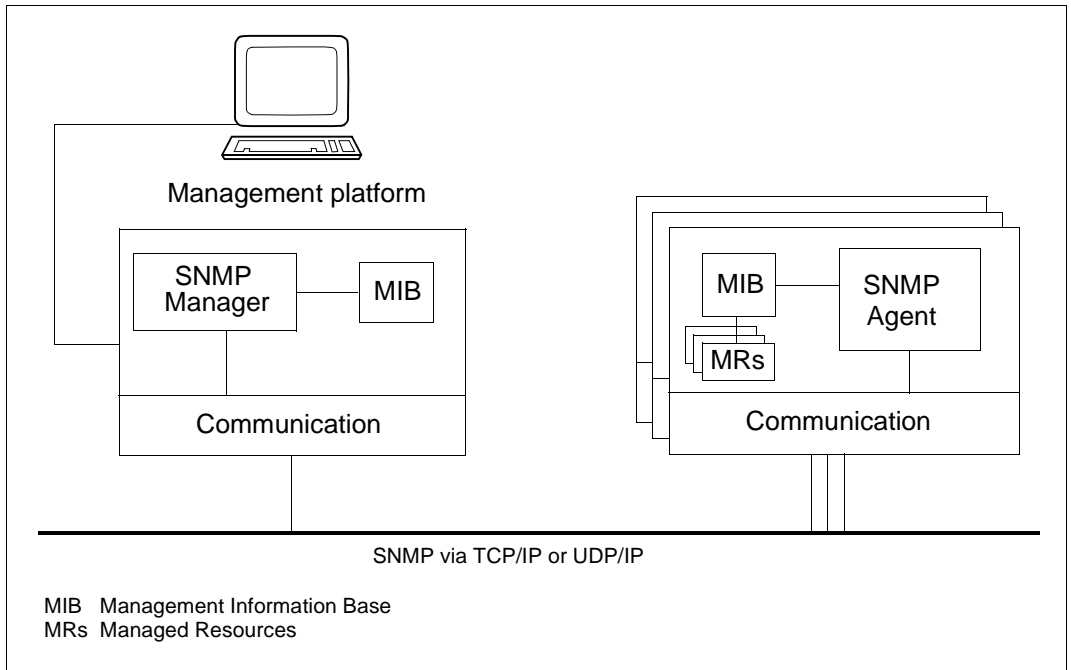


Figure 1: Communication between SNMP manager and agents

## SNMP protocol elements

The information is transported over the network by means of function-dependent SNMP protocol elements. SNMPv1 only requires four different protocol elements for requesting, setting and displaying values that contain the relevant management information (object values). A fifth protocol element, trap, is used by the agents for asynchronous reporting of important events.

Protocol element	Type	Function
GetRequest-PDU	0	Read request from the manager for a specific defined object
GetNextRequest-PDU	1	Read request from the manager for the next (unknown) object
GetResponse-PDU	2	Reply from the agent with the requested values
SetRequest-PDU	3	Write request from the manager to a specific, defined object
Trap-PDU	4	Asynchronous report from the agent when special events occur

SNMPv1 protocol elements

The structure of the actual SNMP message is quite simple. It consists of the SNMP header and PDU (Protocol Data Unit). The SNMP header contains a version ID and the Community Name.

The PDU consists of the field for the PDU type and a list of

- variables to be read (for GetRequest and GetNextRequest) or
- the variable to be set (for SetRequest).

Each variable consists of the name of the object monitored and the associated value. The list of variables that belong to an SNMP message is referred to as variable bindings (short "varbinds").



SNMPv3 provides additional security carried out via security parameters. For details, please refer to [section "Access control"](#).



## 2.2 SNMP management in BS2000 - embedding and functionality

Solutions with different targets are offered for connecting BS2000 to an SNMP management system.

- The products NET-SNMP V5.7 (part of OSD/BC as of V11.0) and SNMP-AGENTS V1.0 offer the capability of integrating BS2000 systems directly into SNMP-based management platforms such as Nagios or Icinga. Some examples are presented in the [chapter “Example for operating the management station”](#). Both NET-SNMP and SNMP-AGENTS allow network, system and application management via an implementation of the SNMP protocol in BS2000 .
- The BCAM, FTP (as part of TCP-IP-AP product) and SESAM/SQL agents supplement the functionality of SNMP-AGENTS, adding more possibilities for BS2000 management. These agents are deployed with their corresponding products. The configuration and functionality of the agents are described in this manual as well.

Figure 2 on the next page gives an overview of the SNMP integration of BS2000.



As a service, Fujitsu Technology Solutions offers the integration into diverse SNMP management systems. For more information; please contact your sales representative.

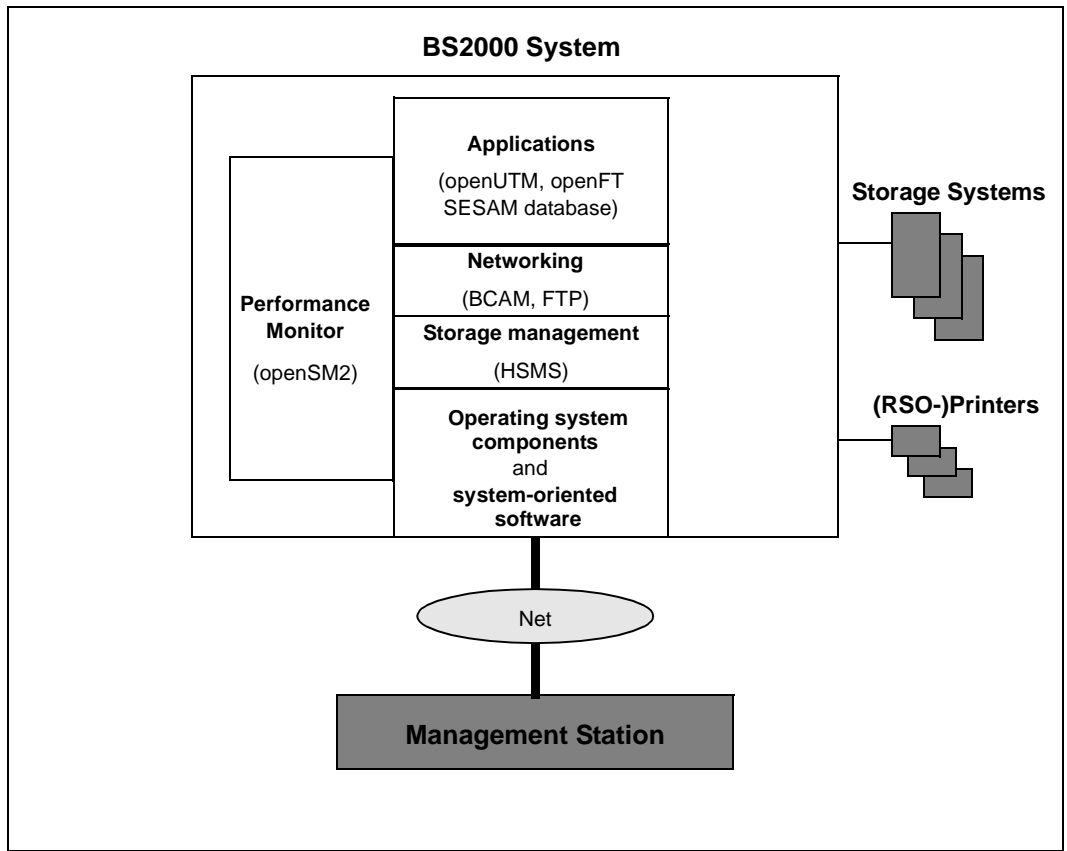


Figure 2: Overview of the systems that SNMP can handle

### 2.2.1 Product structure

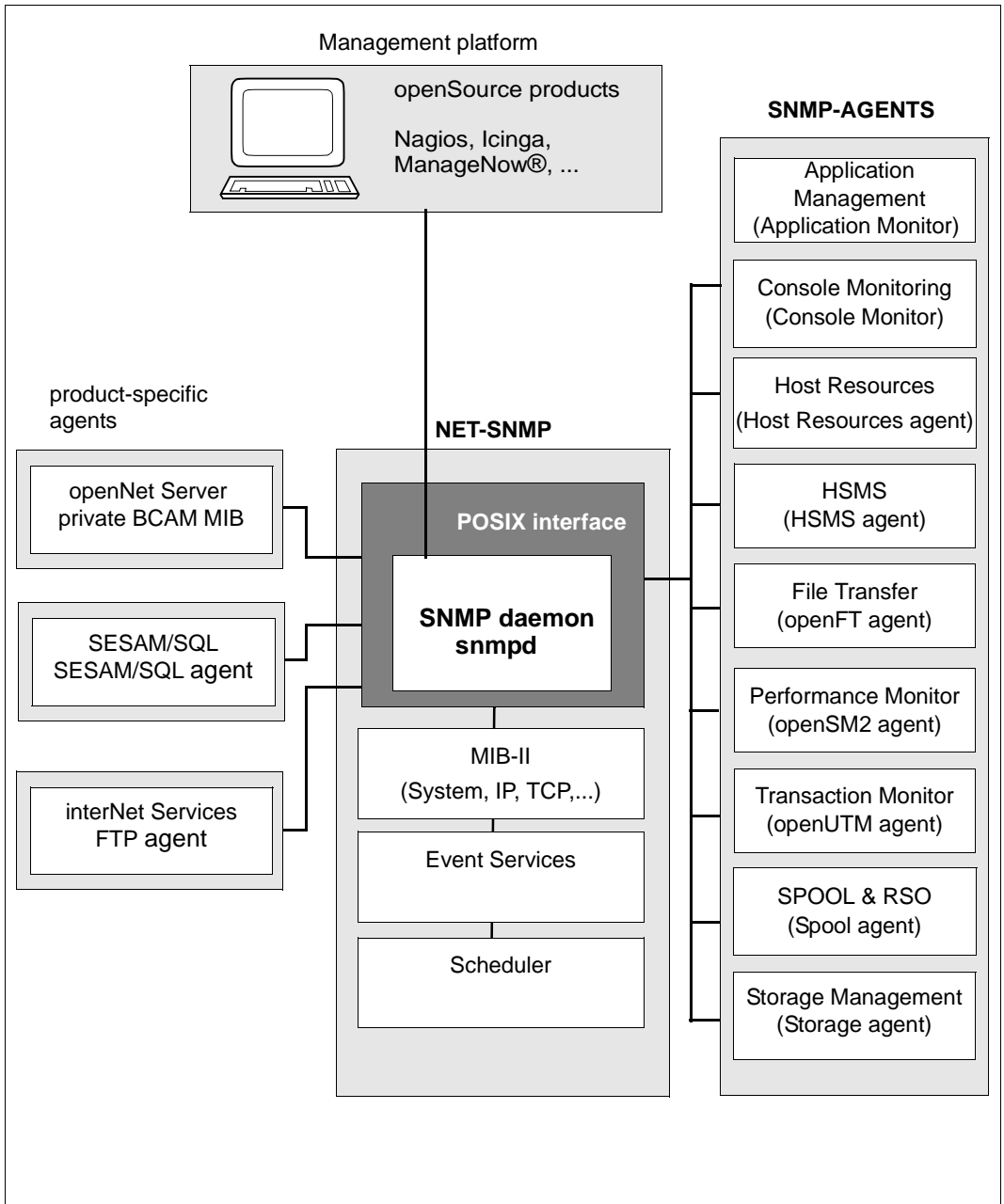


Figure 3: SNMP management structure in BS2000

## NET-SNMP

NET-SNMP is supplied with BS2000 and contains the SNMP daemon *snmpd*, the SNMP trap daemon *snmptrapd* and the SNMP client tools (*snmpwalk*, *snmpget* and *snmpset*). The SNMP daemon also contains the functionality of both Event and Scheduling services, implemented in the Event and Scheduler agents of the previous SNMP versions V6.x.

- The SNMP daemon is the BS2000 communication partner for the management station, which handles the SNMP protocol. It also controls the communication with the agents and offers access to diverse groups of the MIB-II (System, SNMP, Interface, ICMP, IP, TCP, UDP) and the objects of other standardized SNMP MIBs (RFC 2272 - 2275), thereby allowing monitoring of the system and the values relevant to SNMP.
- The Event Services allows you to monitor MIB objects of other agents and perform simple actions, as soon as specific conditions (trigger tests) have been satisfied.
- The Scheduling Services enables modifications to be made to objects periodically or at defined times.

## SNMP-AGENTS

SNMP-AGENTS V1.0 includes a set of agents for BS2000-specific management tasks.

- The Application Monitor agent monitors user applications, BCAM applications, tasks, job variables, BS2000 subsystems and the log files of BS2000, POSIX and NFS. DCAM applications, log files and subsystem states can be monitored cyclically. Logically associated objects in a business process can be grouped together by the Application Monitor agent and monitored either separately or as a group.
- The Console Monitor agent is used for console monitoring. On the one hand, it allows you forward console messages as trap notifications and precisely define the quantity of the messages to be recorded. On the other, you can also issue BS2000 SDF commands from the management station and query the results.
- The Host Resources agent supplies information about the host, devices, pubsets, file systems and the installed software and notifies changes.
- The HSMS agent allows you to read and modify global HSMS data. It also supplies detailed information on HSMS tasks. The scope of the tasks can be restricted by the selection criteria "state" and "origin".
- The openFT (BS2000) agent supplies information about FT system parameters and statistics of the session. It also has the additional functions of starting and stopping the FT, diagnosis control, generating new public keys for encryption and changing the status of an FT partner.
- The performance monitoring agent openSM2 supplies real-time values of monitoring CPU utilization, I/O rates and memory consumption provided by the SM2 subsystem.

- The openUTM agent offers the monitoring and control of selected UTM applications, e.g. information on UTM objects, modifying application properties and system parameters, or terminating an UTM application.
- The agent for spool and print services monitors the SPOOL and RSO devices and supplies information about print jobs.
- The agent for storage management supplies information about pubsets and disks. The agent is also able to monitor the state of the selected pubsets and disks.

### **BCAM, FTP (TCP-IP-AP) and SESAM/SQL agents**

These products are delivering product-specific agents, which are supplementing the functionality of SNMP-AGENTS:

- The BCAM agent supplies information about NEA, ISO and TCP/IP protocols and represents them on the transport system.
- The FTP agent supplies information related to data transfers based on the FTP protocol. It also allows controlling the local FTP daemons by changing their state and parameters.
- The SESAM/SQL agent provides statistical information with regards to the available database handlers. It also provides the SESAM/SQL Performance Monitor measurements output.

## 2.2.2 Structure of the SNMP collection in BS2000

BS2000 is connected to SNMP via a LAN connection that uses TCP/IP protocols. A network management agent that can handle the SNMP protocol elements is installed in BS2000. The functionality of the SNMP agent is split into the SNMP daemon and a number of agents. The advantages of this solution include reliability and user-friendliness with regard to maintenance and modification overhead.

The basis of this solution is the port of the open source software Net-SNMP (released under BSD license) into BS2000. As open source software, Net-SNMP is available on many platforms of different manufacturers.

### 2.2.2.1 SNMP daemon (snmpd)

The current requirements for the SNMP agents in an end system go beyond normal network management. They extend over system and application management up to the management of middleware (transaction systems and databases). Because of the manifold requirements, with large end systems in particular, the desire arises to be able to employ a number of task-specific agents: this is supported by the structuring into master and agents.

The SNMP daemon is hierarchically above the agents. It provides basic functionality, such as handling the SNMP protocol, user access and security management, as well as connectivity to other agents. Note that the SNMP daemon can also be run as a stand-alone agent without being connected to any other agents. The SNMP daemon is therefore also responsible for outputting and setting the values of the various objects found in the MIB-II groups, as well as the ones found in other standardized MIBs (RFC 2272 - 2275).

The option of being able to start and stop agents separately simplifies the updating and use of individual agents without having to run the complete management system down. It also allows interrupt-free management of the remaining system if a component fails, as well as the parallel processing of different agent jobs.

The management station only communicates with the SNMP daemon. The SNMP daemon and agents communicate with each other via an asynchronous message interface. The asynchronous message interface guarantees high master agent performance with job processing because it is not blocked while processing lengthy jobs, but instead can process further SNMP requests in parallel.

### 2.2.2.2 Agents

The agents are only functional if the SNMP daemon is working. In the initialization phase, the agent signs on to the master agent via AgentX socket and passes its MIB to the SNMP daemon.

Agents are event-oriented. The agent goes into a wait loop after initialization. It leaves the loop on arrival of an event that it must process. Requests from the SNMP daemon, timer expiry or the arrival of an agreed signal, are examples of events. The agent goes back into its wait loop after it has processed all existing events.

### 2.2.3 User interface for the SNMP management of BS2000

The standard protocol SNMP enables the connection of BS2000 systems to any management platform that supports SNMP. This is the case for all market-relevant management platforms. The management platforms of the various manufacturers offer a variety of features.

As a service, Fujitsu Technology Solutions offers the integration into diverse SNMP management systems, see [chapter “Example for operating the management station”](#).

### 2.2.4 Parallel operation of SNMP V6.x and NET-SNMP

SNMP V6.x and NET-SNMP can be present simultaneously on the same system. This means that an earlier installation of SNMP V6.x can still be used, where the following should be considered:



Only one instance of the FTP agent (either the old one or new one) can be run at the same time because the old FTP agent and new one communicate with the FTP Server using the same port.

By the default, SNMP binds to UDP port 161 and starts listening for requests on that port. Therefore, if parallel operation of SNMP V6.x and NET-SNMP is expected, one of them should be configured to use a different communication port.

It's possible to use the tools of NET-SNMP, such as *snmpget*, *snmpwalk* and *snmpset* for interacting with SNMP V6.x's daemon *snmpdm* if the SNMP calls use the `-v1` option.

Mixed operation is not possible, i.e. the agents included in SNMP-AGENTS cannot be operated together with SNMP V6.x and the agents included in SNMP V6.x cannot be operated together with NET-SNMP.

## 2.3 Security considerations when using SNMP

This chapter provides notes and recommendations on how to implement secure use of the SNMP-based BS2000 management system with security in mind. It is not however intended as a set of instructions on security analysis or on how to draw up security guidelines. Both of these important subjects go beyond the context of the present manual.

For functional details of how to set the configuration parameters of the SNMP agent in BS2000 with security in mind, see [section “Advanced security options for receiving SNMP requests”](#).

### 2.3.1 Recommendations for general network and system security

With the SNMP protocol you communicate across the internet. However, this exposes the participating systems to the potential attacks and risks associated with the public internet.



#### Recommendation

Put the BS2000 systems and management platform which are to be managed in a subnetwork over which you have exclusive control, for instance, behind a firewall. In this way, you can control access to your systems more easily and more centrally and counter any possible attacks.

The security functions implemented in the SNMP agent are based on the fact that its configuration and program files are only accessible to privileged users, usually an administrator. The installation creates these files with the correct privileges.



#### Recommendation

Check the configuration and program files of the SNMP agent in the BS2000 system at regular intervals to ensure they are only accessible to privileged users.

### 2.3.2 Recommendations for using the SNMP service safely

SNMP is insecure when used naively. The standard configuration file, which is effective after installation comes with a minimum set of rules which are presented as an example for a possible configuration. This is a compromise between safety and comprehensive interoperability in the SNMP network, focusing towards interoperability requirements.

Avoid live operation with a minimum configuration of the BS2000 system. If you change the configuration on the BS2000 system being managed, please adapt the corresponding settings on the management platform.



**Recommendation**

Change the minimum configuration of the SNMP agent and the corresponding settings on the management platform in accordance with the guidelines of your security manual.

You should be particularly careful from a security point of view with the following configuration parameters:

- Community strings for receiving SNMP requests (SNMP protocol version 1 and 2c)
- Community strings and access control of MIB objects
- Advanced security options for receiving SNMP requests (SNMPv3)
- Community strings and sender addresses
- Recipient's addresses for SNMP traps

**2.3.2.1 Community strings for receiving SNMP requests**

In SNMP protocol version 1 and 2c, a "community" denotes a group comprising one or more management platforms and several SNMP agents handled by these platforms.

Every community is identified by a community string. The community string is a non-encrypted component of every SNMP request and identifies the sender of the request as a member of the community concerned. Authorization for a read or write request which a management platform sends to an SNMP agent is controlled with this community string.

The community string makes a simple authentication mechanism available in SNMP. Management platforms and SNMP agents may only communicate with one another if they belong to the same community: the SNMP agent will only accept SNMP requests from management platforms whose community strings are known to it, i.e. preconfigured.

Since the community string is sent in non-encrypted form with the SNMP message, it is always at a risk of being used without authorization. This can be problematic for using SNMP with security in mind. On the other hand, most communities use the preset community string "public" in any case.

**Recommendation**

Select suitable communities corresponding to the organization of your systems and operations and assign suitable community strings to them. Change the community string in accordance with the guidelines of your manual: in a similar way, for instance, as for passwords. Note that you must modify the community string in all participating systems in the community.

**Recommendation**

If the environment of your SNMP agents and management platform(s) allows you to do so, use the user-specific authentication available with the SNMPv3 protocol.

**2.3.2.2 Advanced security options for receiving SNMP requests**

SNMPv3 introduces advanced security model - USM (User-based security model) - which contains a list of users and their attributes. The USM is described by RFC 2574.

Each user has a name (called a *securityName*) an authentication type (*authProtocol*) and a privacy type (*privProtocol*) as well as associated keys for each of these (*authKey* and *privKey*).

Authentication is performed by using a user's *authKey* to sign the message being sent. The *authProtocol* can be either MD5 or SHA at this time. *authKeys* (and *privKeys*) are generated from a passphrase that must be at least 8 characters in length.

Authentication is performed by using a user's *privKey* to encrypt the data portion the message being sent. The *privProtocol* can be either AES or DES.

Messages can be sent unauthenticated, authenticated, or authenticated and encrypted by setting the *securityLevel* to use.

**2.3.2.3 Community strings and controlling access to MIB objects**

Community strings can be given the access rights "read-only", "read-write" etc. In accordance with these access rights, SNMP requests containing this community string may read objects defined as "read-only" and read and change objects defined as "read-write". If there are no further precautions, *all* accessible objects can be read or *all* changeable objects can be read and changed.

**Recommendation**

Use the option of assigning selective read or write privileges to specific community strings, for:

- MIB branches
- objects
- object instances (in tables)

**Recommendation**

If the environment of your SNMP agents and management platform(s) allows you to do so, use the user-specific authorization available in the SNMPv3 protocol.

### 2.3.2.4 Community strings and sender addresses

You can explicitly preconfigure the IP addresses of the authorized management platforms. In this way, you force the SNMP agent to only accept SNMP requests from these systems.

The management platforms must have fixed IP addresses to do this. It is not possible to dynamically assign them with the Dynamic Host Configuration Protocol (DHCP).



#### Recommendation

Check the sender addresses of the management platforms by configuring their IP addresses in the SNMP agent.

### 2.3.2.5 Recipient's addresses for SNMP traps

You can explicitly predefine the IP addresses of the authorized management platforms to which an SNMP agent is to send SNMP traps. The management platforms must have constant IP addresses to do this. It is not possible to dynamically assign them with the Dynamic Host Configuration Protocol (DHCP).



#### Recommendation

Predefine the receiver's addresses of the management platforms to receive SNMP traps. To do this, configure the IP addresses of these management platforms in the SNMP agent.

### 2.3.2.6 Community string for SNMP traps

You can configure the community string which, an SNMP agent sends to the management platform as part of an SNMP trap. The management platform will then only accept SNMP traps with this community string.



#### Recommendation

Select suitable communities corresponding to the organization of your systems and operations. Change the community string in accordance with the guidelines of your manual: in a similar way, for instance, as for passwords. Note that you must change the community string in all participating systems in the community.



---

## 3 Installation and configuration

BS2000-SNMP management consists of the following products for use on the BS2000 system:

- NET-SNMP V5.7 (is a delivery component of BS2000 OSD/BC as of V11.0)
- SNMP-AGENTS V1.0
- BCAM V24.0
- TCP-IP-AP V5.3 (FTP agent)
- SESAM\SQL V9.1

### 3.1 Software requirements

#### **Common software requirements for NET-SNMP, SNMP-AGENTS, BCAM, TCP-IP-AP, SESAM/SQL**

- BS2000 OSD/BC V11.0 or higher with appropriate software configuration
- POSIX A45 or higher.

#### **Software requirements for NET-SNMP V5.7**

- openNet Server V4.0 or higher

#### **Software requirements for SNMP-AGENTS V1.0**

- NET-SNMP V5.7 or higher
- SDF-P-BASYS V2.5 or higher

When using single agents the following is required:

- JV V15.1 or higher (Application Monitor agent)
- openFT (BS2000) V12.1 or higher (openFT agent)
- SPOOL V4.9 or higher (Spool agent)

- HSMS V11.0 or higher (HSMS agent)
- SM2 V20.0 or higher (openSM2 agent)
- openUTM V6.4 or higher (openUTM agent)

More information can be found in the release notices of BS2000 OSD/BC and the corresponding products.

## 3.2 Installation in BS2000 OSD/BC

NET-SNMP is a component of the BS2000 OSD/BC operating system. The necessary files are located on the BS2000 system after BS2000 OSD/BC has been installed. To make NET-SNMP executable, it still requires a packet installation in POSIX. This can be performed either manually or via an automatic POSIX package installation using IMON.

To complete the installation manually, please refer to [Installing products manually](#).

As an optional products, SNMP-AGENTS, TCP-IP-AP (part of interNet Services), BCAM and SESAM/SQL are not installed on a BS2000 system by default.

The installation is carried out by the installation monitor IMON. Where necessary, the IMON installation performs BS2000-specific jobs such as creation of subsystem catalog entries, POSIX installation, etc.

For more information, please refer to the latest IMON manual.



- It must be noted that an entry for the SNMPAGT subsystem is generated in the system catalog.
- Deleting the SINLIB after installation leads to errors as the agents also require the SINLIB during operation.

To complete the installation for one of the SNMP products manually, please refer to [section “Installing products manually”](#).



Using IMON is the preferable way to install products. It is not recommended to perform manual installation for any of the products.

### 3.2.1 Installation defaults

All SNMP products are facilitating *syslog* for logging. Logging options could be changed globally by editing *snmp.conf* or individually for each daemon/agent by editing the appropriate rc file (auto start/stop script passing command line arguments):

- The rc files located in *etc/rc2.d* are invoked during startup of the POSIX subsystem and follow the standard naming convention of starting with the letter "S".
- The rc files located in *etc/rc0.d* are invoked during shutdown of the POSIX subsystem and follow the standard naming convention of starting with the letter "K".

The */usr/share/snmp/mibs* folder is used for storing the MIB files of installed products.

*/etc/snmp* is used for storing configuration files and backed up starting rc files.

The following table summarizes the delivery scope of the SNMP-related products.

<b>Product</b>	<b>Path to the binaries</b>	<b>Binary</b>	<b>Config files</b>	<b>rc scripts</b>
NET-SNMP	/opt/net-snmp	snmpd snmptrapd snmpwalk snmpget snmpset	snmpd.conf snmp.conf snmptrapd.conf	S90net-snmp K11net-snmp
SNMP-AGENTS	/opt/snmp-agents	appMonAgent consoleAgent openFTAgent openSM2Agent spoolAgent storageAgent hostAgent hsmsAgent utmAgent	none	S91snmp-agents K11snmp-agents
TCP-IP-AP		ftpAgent	none	S91snmpftp K11snmpftp
BCAM		bcamAgent	none	S91snmpbcam K11snmpbcam
SESAM/SQL		sesAgent	none	S91snmpsesam K11snmpsesam



### 3.2.2 Delivery scope of NET-SNMP

The following binaries are shipped with NET-SNMP:

- snmpd (SNMP daemon)
- snmptrapd (SNMP trap daemon)
- **SNMP tools** snmpwalk, snmpget, snmpset

In addition, the following example configuration files are supplied for the daemons and tools:

- snmpd.conf: configuration file for the SNMP daemon snmpd
- snmptrapd.conf: configuration file for the trap daemon snmptrapd
- snmp.conf: general configuration file for SNMP daemons and tools



The rc file `/etc/rc2.d/S90net-snmp` can also be used for restarting (reconfiguring) the SNMP daemon that is started during POSIX startup.

This can be done by issuing the following command under privileged user:

```
/etc/rc2.d/S90net-snmp restart
```

### 3.2.3 Installing products manually

Installation is carried out via the POSIX installation program, which is started by the START-POSIX-INSTALLATION command. For this, the POSIX must have been started. For more details, please refer to the POSIX (BS2000) manual "Basics for Users and System Administrators".

For convenience, during installation informational messages are outputted.

#### *NET-SNMP output example*

```
New configuration file was created </etc/snmp/snmp.conf.new>.
Please, review it and apply to snmp.conf if needed.
New configuration file was created </etc/snmp/snmpd.conf.new>.
Please, review it and apply to snmpd.conf if needed.
New configuration file was created </etc/snmp/snmptrapd.conf.new>.
Please, review it and apply to snmptrapd.conf if needed.
```

```
NET-SNMP v057 is INSTALLED:
Binaries are located in /opt/net-snmp with symlinks to the /usr/bin
Daemons binary links are in /usr/sbin
PATH for the config files: /etc/snmp
PATH for the MIB files: /usr/share/snmp/mibs
```

### 3.2.4 Uninstallation

SNMP products are also uninstalled via IMON or manually by means of POSIX installation program (START-POSIX-INSTALLATION).

The following items are deleted during uninstallation:

- Symbolic links in */usr/sbin* and/or */usr/bin*
- The corresponding object files in */opt/net-snmp* and/or */opt/snmp-agents*
- The */opt/net-snmp* and/or */opt/snmp-agents* directory(ies)
- The rc scripts are deleted, but starting scripts (from */etc/rc2.d* only) are backed up in order to keep the user's configuration. The backed up rc-scripts can be found in the */etc/snmp* folder under name of type *ORIGINAL\_NAME.bkp*.

Neither the MIB files nor the configuration files are deleted.

### 3.3 SNMP configuration in BS2000

The work to be carried out in BS2000 is described in the sections listed below. For general recommendations, please refer to [section “Recommendations for using the SNMP service safely”](#).

Configuration can be useful for controlling the fundamental nature of all of the SNMP applications. This could be achieved by using one of the available configuration methods in the following order of preference:

- Command line arguments
- Environment variables (MIBS and MIBDIRS)
- Configuration files with extension *.conf* or *.local.conf* (will be read the last) from the following directories:
  - /etc/snmp (searched the first)
  - /usr/local/share/snmp
  - /usr/local/lib/snmp
  - ~/.snmp
  - /var/net-snmp (searched the latest)

There are a couple of the SNMP configuration files available for different parts of NET-SNMP:

- *<NAME\_OF\_BINARY>.conf* or *<NAME\_OF\_BINARY>.local.conf*  
controls parameters and capabilities of the specific binary.

*Example:*

```
snmpd -> snmpd.conf
utmAgent -> utmAgent.conf
snmpwalk -> snmpwalk.conf
```

- *snmp.conf* or *snmp.local.conf*  
NET-SNMP general configuration file.
- *agentx.conf* or *agentx.local.conf*  
master/agent configuration for AgentX
- *snmpapp.conf* or *snmpapp.local.conf*  
SNMP tools configuration (*snmpwalk*, *snmpget*...)

Each application may use multiple methods of configuration. In fact, most applications understand how to read the contents of the *snmp.conf* files.

Note, however, that configuration directives understood in one file may not be understood in another file. Applications support a *-H* switch on the command line that will list the configuration files it will look for and the directives in each one that the corresponding understands.

### 3.3.1 Listening addresses in BS2000

By default, *snmpd* listens for incoming SNMP requests on UDP port 161 on all IPv4 addresses. However, it is possible to modify this behavior by specifying one or more listening addresses as arguments to *snmpd*. A listening address takes the form:

```
[<transport-specifier>:]<transport-address>
```

At its simplest, a listening address may consist only of a port number, in which case *snmpd* listens on that UDP port on all IPv4 interfaces.

Otherwise, the *<transport-address>* part of the specification is parsed according to the following:

*transport-address format*

udp (default)	hostname[:port] or IPv4-address[:port]
tcp	hostname[:port] or IPv4-address[:port]
unix	pathname
udp6 or udpv6 or udpipv6	hostname[:port] or IPv6-address[:port]
tcp6 or tcpv6 or tcpipv6	hostname[:port] or IPv6-address[:port]

Note that *<transport-specifier>* strings are not case-insensitive so that, for example, "tcp" and "TCP" are equivalent. Here are some examples, along with their interpretation:

127.0.0.1:161

listen on UDP port 161, but only on the loopback interface. This prevents *snmpd* being queried remotely.

TCP:1161

listen on TCP port 1161 on all IPv4 interfaces.

unix:/tmp/local-agent

listen on the Unix domain socket */tmp/local* agent.

/tmp/local-agent

is identical to the previous specification, since the Unix domain is assumed if the first character of the *<transport-address>* is *'/'*.

udp6:10161

listen on port 10161 on all IPv6 interfaces.

Note that not all the transport domains listed above will always be available; for instance, hosts with noIPv6 support will not be able to use udp6 transport addresses, and attempts to do so will result in the error "Error opening specified endpoint".

### 3.3.2 SNMP general configuration (snmp.conf)

#### 3.3.2.1 Client behavior

defDomain application domain

The transport domain that should be used for a certain application type unless something else is specified.

defTarget application domain target

The target that should be used for connections to a certain application if the connection should be in a specific domain.

defaultPort PORT

defines the default UDP port that client SNMP applications will attempt to connect to. This can be overridden by explicitly including a port number in the AGENT specification. See the *snmpcmd(1)* manual page for more details. If not specified, the default value for this token is 161.

defVersion (1|2c|3)

defines the default version of SNMP to use. This can be overridden using the *-v* option.

defCommunity STRING

defines the default community to use for SNMPv1 and SNMPv2c requests. This can be overridden using the *-c* option.

alias NAME DEFINITION

creates an aliased tied to NAME for a given transport definition. The alias can be referred to using an alias: NAME. E.g., a line of "alias here udp:127.0.0.1:6161" would allow you to use a destination host of "alias:here" instead of "udp:127.0.0.1:6161". This becomes more useful with complex transport addresses involving IPv6 addresses, etc.

dumpPacket yes

defines whether to display a hexadecimal dump of the raw SNMP requests sent and received by the application. This is equivalent to the *-d* option.

doDebugging (1|0)

turns on debugging for all applications run if set to 1.

debugTokens TOKEN[,TOKEN...]

defines the debugging tokens that should be turned on when *doDebugging* is set. This is equivalent to the *-D* option.

16bitIDs yes

restricts requestIDs, etc to 16-bit values.

The SNMP specifications define these ID fields as 32-bit quantities, and the Net-SNMP library typically initializes them to random values for security. However certain (broken) agents cannot handle IDvalues greater than  $2^{16}$ . This option allows interoperability with such agents.

clientaddr [<transport-specifier>:]<transport-address>

specifies the source address to be used by command line applications when sending SNMP requests. This value is also used by *snmpd* when generating notifications.

clientSendBuf INTEGER

is similar to *clientRecvBuf*, but applies to the size of the buffer used when sending SNMP requests.

noRangeCheck yes

disables the validation of varbind values against the MIB definition for the relevant OID.

noTokenWarnings

disables warnings about unknown config file tokens.

reverseEncodeBER (1|yes|true|0|no|false)

controls how the encoding of SNMP requests is handled.

The default behavior is to encode packets starting from the end of the PDU and working backwards. This directive can be used to disable this behavior, and build the encoded request in the (more obvious) forward direction.

### 3.3.2.2 SNMPv3 settings

`defSecurityName` STRING

defines the default security name to use for SNMPv3 requests. This can be overridden using the `-u` option.

`defSecurityLevel` `noAuthNoPriv|authNoPriv|authPriv`

defines the default security level to use for SNMPv3 requests. This can be overridden using the `-l` option.

If not specified, the default value for this token is `noAuthNoPriv`.



`authPriv` is only available if the software has been compiled to use the OpenSSL libraries.

`defPassphrase` STRING

`defAuthPassphrase` STRING

`defPrivPassphrase` STRING

define the default authentication and privacy passphrases to use for SNMPv3 requests. These can be overridden using the `-A` and `-X` options respectively.

The `defPassphrase` value will be used for the authentication and/or privacy passphrases if either of the other directives are not specified.

`defAuthType` `MD5|SHA`

`defPrivType` `DES|AES`

define the default authentication and privacy protocols to use for SNMPv3 requests. These can be overridden using the `-a` and `-x` options respectively.

If not specified, SNMPv3 requests will default to MD5 authentication and DES encryption.



If the software has not been compiled to use the OpenSSL libraries, then only MD5 authentication is supported. Neither SHA authentication nor any form of encryption will be available.

`defContext` STRING

defines the default context to use for SNMPv3 requests. This can be overridden using the `-n` option.

If not specified, the default value for this token is the default context (i.e. the empty string `""`).

`defSecurityModel` STRING

defines the security model to use for SNMPv3 requests. The default value is `usm`, which is the only widely used security model for SNMPv3.

```
defAuthMasterKey 0xHEXSTRING
defPrivMasterKey 0xHEXSTRING
defAuthLocalizedKey 0xHEXSTRING
defPrivLocalizedKey 0xHEXSTRING
```

define the (hexadecimal) keys to be used for SNMPv3 secure communications.

SNMPv3 keys are frequently derived from a passphrase, as discussed in the *defPassphrase* section above. However for improved security a truly random key can be generated and used instead (which would normally has better entropy than a password unless it is amazingly long). The directives are equivalent to the short-form command line options *-3m*, *-3M*, *-3k*, and *-3K*.

Localized keys are master keys which have been converted to a unique key which is only suitable for on particular SNMP engine (agent). The length of the key needs to be appropriate for the authentication or encryption type being used:

auth keys: MD5=16 bytes, SHA1=20 bytes;

priv keys: DES=16bytes (8 bytes of which is used as an IV and not a key), and AES=16 bytes.

### 3.3.2.3 Server behavior

```
persistentDir DIRECTORY
```

defines the directory where *snmpd* and *snmptrapd* store persistent configuration settings.

If not specified, the persistent directory defaults to */var/net-snmp*.

```
noPersistentLoad yes
```

```
noPersistentSave yes
```

disable the loading and saving of persistent configuration information.



This will break SNMPv3 operations (and other behavior that relies on changes persisting across application restart). Use with care!

```
tempFilePattern PATTERN
```

defines a filename template for creating temporary files, for handling input to and output from external shell commands. Used by the *mkstemp()* and *mktemp()* functions.

If not specified, the default pattern is */tmp/snmpdXXXXXX*.



### 3.3.2.4 MIB handling

mibdirs DIRLIST

specifies a list of directories to search for MIB files. Note that this value can be overridden by the MIBDIRS environment variable, and the *-M* option.

mibs MIBLIST

specifies a list of MIB modules (not files) that should be loaded. Note that this list can be overridden by the MIBS environment variable, and the *-m* option.

mibfile FILE

specifies a (single) MIB file to load, in addition to the list read from the *mibs* token (or equivalent configuration). Note that this value can be overridden by the MIBFILES environment variable.

showMibErrors (1|yes|true|0|no|false)

whether to display MIB parsing errors.

commentToEOL (1|yes|true|0|no|false)

whether MIB parsing should be strict about comment termination. Many MIB writers assume that ASN.1 comments extend to the end of the text line, rather than being terminated by the next "--" token. This token can be used to accept such (strictly incorrect) MIBs. Note that this directive was previous (mis-)named *strictCommentTerm*, but with the reverse behavior from that implied by the name. This earlier token is still accepted for backwards compatibility.

mibAllowUnderline (1|yes|true|0|no|false)

whether to allow underline characters in MIB object names and enumeration values. This token can be used to accept such (strictly incorrect) MIBs.

mibWarningLevel INTEGER

the minimum warning level of the warnings printed by the MIB parser.

### 3.3.2.5 Output configuration

Most of options from this paragraph can also be configured via command line arguments.



Those options are relevant only for binaries from NET-SNMP product.

logTimestamp (1|yes|true|0|no|false)

Whether the commands should log timestamps with their error/message logging or not. Note that output will not look as pretty with timestamps if the source code that is doing the logging does incremental logging of messages that are not line buffered before being passed to the logging routines. This option is only used when file logging is active.

**printNumericEnums (1|yes|true|0|no|false)**

Equivalent to *-Oe*. Removes the symbolic labels from enumeration values:

```
$ snmpget -c public -v 1 localhost ipForwarding.0
IP-MIB::ipForwarding.0 = INTEGER: forwarding(1)
$ snmpget -c public -v 1 -Oe localhost ipForwarding.0
IP-MIB::ipForwarding.0 = INTEGER: 1
```

**printNumericOids (1|yes|true|0|no|false)**

Equivalent to *-On*. Displays the OID numerically:

```
.1.3.6.1.2.1.1.3.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

**dontBreakdownOids (1|yes|true|0|no|false)**

Equivalent to *-Ob*. Display table indexes numerically, rather than trying to interpret the instance sub-identifiers as string or OID values:

```
$ snmpgetnext -c public -v 1 localhost vacmSecurityModel
SNMP-VIEW-BASED-ACM-MIB::vacmSecurityModel.0."wes" = xxx
$ snmpgetnext -c public -v 1 -Ob localhost vacmSecurityModel
SNMP-VIEW-BASED-ACM-MIB::vacmSecurityModel.0.3.119.101.115 = xxx
```

**escapeQuotes (1|yes|true|0|no|false)**

Equivalent to *-OE*. Modifies index strings to escape the quote characters:

```
$ snmpgetnext -c public -v 1 localhost vacmSecurityModel
SNMP-VIEW-BASED-ACM-MIB::vacmSecurityModel.0."wes" = xxx
$ snmpgetnext -c public -v 1 -OE localhost vacmSecurityModel
SNMP-VIEW-BASED-ACM-MIB::vacmSecurityModel.0.\"wes\" = xxx
```

This allows the output to be reused in shell commands.

**quickPrinting (1|yes|true|0|no|false)**

Equivalent to *-Oq*. Removes the equal sign and type information when displaying varbind values:

```
SNMPv2-MIB::sysUpTime.0 1:15:09:27.63
```

**printValueOnly (1|yes|true|0|no|false)**

Equivalent to *-Ov*. Display the varbind value only, not the OID:

```
$ snmpget -c public -v 1 -Oe localhost ipForwarding.0
INTEGER: forwarding(1)
```

**dontPrintUnits (1|yes|true|0|no|false)**

Equivalent to *-OU*. Do not print the UNITS suffix at the end of the value.

**numericTimeticks (1|yes|true|0|no|false)**

Equivalent to *-Ot*. Display TimeTicks values as raw numbers:

```
SNMPv2-MIB::sysUpTime.0 = 14096763
```

`printHexText (1|yes|true|0|no|false)`

Equivalent to `-OT`. If values are printed as Hex strings, display a printable version as well.

`hexOutputLength` integer

Specifies where to break up the output of hexadecimal strings. Set to 0 to disable line breaks. Defaults to 16.

`suffixPrinting (0|1|2)`

The value 1 is equivalent to `-Os`; displays the MIB object name (plus any instance or other sub identifiers):

```
sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

The value 2 is equivalent to `-OS`; displays the name of the MIB, as well as the object name:

```
SNMPv2-MIB::sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

This is the default OID output format.

`oidOutputFormat (1|2|3|4|5|6)`

Maps `-O` options as follows:

`-Os=1`, `-OS=2`, `-Of=3`, `-On=4`, `-Ou=5`.

The value 6 has no matching `-O` option. It suppresses output.

'-Of'

Include the full list of MIB objects when displaying an OID:

```
iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0
```

'-Ou'

display the OID in the traditional UCD-style (inherited from the original CMU code). That means removing a series of "standard" prefixes from the OID, and displaying the remaining list of MIB object names (plus any other sub identifiers):

```
system.sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

`extendedIndex (1|yes|true|0|no|false)`

Equivalent to `-OX`. Display table indexes in a more "program like" output, imitating a traditional array-style index format:

```
$ snmpgetnext -c public -v 1 localhost ipv6RouteTable
```

```
IPv6-MIB::ipv6RouteIfIndex.63.254.1.0.255.0.0.0.0.0.0.0.0.0.0.0.0.64.1 =  
INTEGER: 2
```

```
$ snmpgetnext -c public -v 1 -OE localhost ipv6RouteTable
```

```
IPv6-MIB::ipv6RouteIfIndex[3ffe:100:ff00:0:0:0:0:0][64][1] = INTEGER: 2
```

`noDisplayHint (1|yes|true|0|no|false)`

Disables the use of `DISPLAY-HINT` information when parsing indices and values to set.

### 3.3.3 AgentX configuration (agentx.conf)

Relevant tokens are:

agentxsocket

AgentX bind address

agentxperms

AgentX socket permissions

agentxRetries

AgentX Retries

agentxTimeout

AgentX Timeout (seconds)

The description could be found in [section "AgentX options"](#).

### 3.3.4 Command line arguments

Configuration directives from configuration files (relative to the given binary) could be passed via command line too. For this specify it like this:

```
--CONF_DIRECTIVE="VALUE1[ VALUE2...]"
```

*Example:*

```
snmpd --rwcommunity="public "
```

There are also command line arguments, which are binary specific and can be retrieved via calling the binary with `--help` directive. Some of them could not be configured via configuration files. Next table will explore some important options, which could be used by master, agents and SNMP tools.

Option	Description
<i>General options</i>	
-m MIBLIST	use MIBLIST instead of the default MIB list
-M DIRLIST	use DIRLIST as the list of locations to look for MIBs
-d	Dump (in hexadecimal) the raw SNMP packets sent and received.
-D[TOKEN[,...]]	Turn on debugging output for the given TOKEN(s). For specific available tokens, please refer to <a href="#">section "Debug options"</a> .
-H	Display a list of configuration file directives understood by the command and then exit.
-f	do not fork from the shell
-p FILE	store process id in FILE
-L	Logging options
-Le	Log messages to the standard error stream.
-Lf FILE	Log messages to the specified file.
-Lo	Log messages to the standard output stream.
-Ls FACILITY	Log messages via <i>syslog</i> , using the specified facility: 'd' for LOG_DAEMON, 'u' for LOG_USER, '0'-'7' for LOG_LOCAL0 - LOG_LOCAL7)
-LE pri	Log messages of priority 'pri' and above to standard error.
-LS pri	similar as previous, but logging to syslog
-LE p1-p2	Log messages with priority between 'p1' and 'p2' (inclusive) to standard error.
-LS p1-p2	similar as previous, but logging to <i>syslog</i>
<i>Daemons specific options</i>	
-c FILE[,...]	read FILE(s) as configuration file(s)
-C	do not read the default configuration files

Option	Description
-x ADDRESS	use ADDRESS as AgentX address
-X	run as an AgentX master
<i>snmpd specific options</i>	
-a	log addresses
-q	print information in a more parsable format
<i>snmptrapd options</i>	
-a	ignore authentication failure traps
-n	use numeric addresses instead of attempting hostname lookups (no DNS)
-t	Prevent traps from being logged to <i>syslog</i>

For *-LF* and *-LS* the priority specification comes before the file or facility token. The priorities recognized are:

Priority levels	Means
0 or !	LOG_EMERG
1 or a	LOG_ALERT
2 or c	LOG_CRIT
3 or e	LOG_ERR
4 or w	LOG_WARNING
5 or n	LOG_NOTICE
6 or i	LOG_INFO
7 or d	LOG_DEBUG

By default, output is logged to the *syslog* at a priority level of LOG\_INFO.

## 3.4 Configuring NET-SNMP

### 3.4.1 Configuring SNMP daemon `snmpd` (`snmpd.conf`)

The Net-SNMP daemon uses one or more configuration files to control its operation and the management information provided. For instance (`snmpd`) knows how to understand configuration directives in both the `snmpd.conf` and the `snmp.conf` files

Next chapters will provide general information about daemon's configuration carried out via `snmpd.conf` file. As NET-SNMP is an open source project, more information and examples could be found at the informational web-services and man pages.



Some of the open source NET-SNMP functionality is not available on BS2000, therefore it is recommend to check information from WEB with current manual.

#### 3.4.1.1 Agent behavior

Although most configuration directives are concerned with the MIB information supplied by the agent, there are a handful of directives that control the behavior of `snmpd` considered simply as a daemon providing a network service.

`agentaddress [<transport-specifier>:]<transport-address>[,...]`

defines a list of listening addresses, on which to receive incoming SNMP requests. See [section "Listening addresses in BS2000"](#) details.

The default behavior is to listen on UDP port 161 on all IPv4 interfaces.



You can specify more than ore listening address.

Due to the technical reasons IPv6 ports cannot match IPv4 ports, e.g.:

`agentaddress udp:161,tcp:161 - will work`

`agentaddress udp:161,udp6:161 - will not work`

`agentaddress udp:161,udp6:163 - will work`

`agentgroup {GROUP|#GID}`

changes to the specified group after opening the listening port(s). This may refer to a group by name (GROUP), or a numeric group ID starting with '#' (#GID).

`agentuser {USER|#UID}`

changes to the specified user after opening the listening port(s). This may refer to a user by name (USER), or a numeric user ID starting with '#' (#UID).

`leave_pidfile yes`

instructs the agent to not remove its *pid* file on shutdown. Equivalent to specifying `-U` on the command line.

`maxGetbulkRepeats NUM`

Sets the maximum number of responses allowed for a single variable in a *getbulk* request. Set to 0 to enable the default and set it to -1 to enable unlimited. Because memory is allocated ahead of time, sitting this to unlimited is not considered safe if your user population cannot be trusted. A repeat number greater than this will be truncated to this value.

This is set by default to -1.

`maxGetbulkResponses NUM`

Sets the maximum number of responses allowed for a *getbulk* request. This is set by default to 100. Set to 0 to enable the default and set it to -1 to enable unlimited. Because memory is allocated ahead of time, setting this to unlimited is not considered safe if your user population cannot be trusted.

In general, the total number of responses will not be allowed to exceed the *maxGetbulkResponses* number and the total number returned will be an integer multiple of the number of variables requested times the calculated number of repeats allow to fit below this number.

Also note that processing of *maxGetbulkRepeats* is handled first.

### 3.4.1.2 AgentX options

The Net-SNMP and supplementary products support the AgentX protocol (RFC 2741) in both master and agent roles. Use of this mechanism requires that the daemon has agentx module explicitly enabled (e.g. via the *snmpd.conf* file).

There are two directives specifically relevant to running as an AgentX master:

`master agentx`

will enable the AgentX functionality and cause the agent to start listening for incoming AgentX registrations. This can also be activated by specifying the `-x` command line option (to specify an alternative listening socket).

`agentXPerms SOCKPERMS [DIRPERMS [USER|UID [GROUP|GID]]]`

Defines the permissions and ownership of the AgentX Unix Domain socket, and the parent directories of this socket. SOCKPERMS and DIRPERMS must be octal digits (see *chmod(1)*). By default this socket will only be accessible to agents which have the same userid as the agent.



There is one directive specifically relevant to running as an AgentX agent:

`agentXPingInterval NUM`

will make the agent try and reconnect every NUM seconds to the master if it ever becomes (or starts) disconnected.

The remaining directives are relevant to both AgentX master and agents:

`agentXSocket [<transport-specifier>:]<transport-address>[,...]`

defines the address the master agent listens at, or the agent should connect to. The default is the Unix Domain socket `/var/agentx/master`. Another common alternative is `tcp:localhost:705`.



Specifying an AgentX socket does not automatically enable AgentX functionality (unlike the `-x` command line option).

`agentXTimeout NUM`

defines the timeout period (NUM seconds) for an AgentX request. Default is 1 second.

`agentXRetries NUM`

defines the number of retries for an AgentX request. Default is 5 retries.

### 3.4.1.3 SNMPv3 configuration

SNMPv3 requires an SNMP agent to define a unique "engine ID" in order to respond to SNMPv3 requests. This ID will normally be determined automatically, using two reasonably non-predictable values:

- a (pseudo-)random number and
- the current time in seconds.

This is the recommended approach.

However the capacity exists to define the *engineID* in other ways:

`engineID STRING`

specifies that the *engineID* should be built from the given text STRING.

`engineIDType 1|2|3`

specifies that the *engineID* should be built from the IPv4 address (1), IPv6 address (2) or MAC address (3). Note that changing the IP address (or switching the network interface card) may cause problems.

`engineDNic INTERFACE`

defines which interface to use when determining the MAC address. If *engineID* Type 3 is not specified, then this directive has no effect.

The default is to use *eth0*.

### 3.4.1.4 SNMPv3 authentication

SNMPv3 was originally defined using the User-Based Security Model (USM), which contains a private list of users and keys specific to the SNMPv3 protocol.

To use the USM based SNMPv3-specific users, you'll need to create them explicitly:

```
createUser [-e ENGINEID] username (MD5|SHA) authpassphrase [DES|AES]
           [privpassphrase]
```

MD5 and SHA are the authentication types to use. DES and AES are the privacy protocols to use. If the privacy passphrase is not specified, it is assumed to be the same as the authentication passphrase. Note that the users created will be useless unless they are also added to the VACM access control tables, see [section "Access control"](#).



#### CAUTION!

The minimum passphrase length is 8 characters.

### 3.4.1.5 Access control

*snmpd* supports the View-Based Access Control Model (VACM) as defined in RFC 2575, to control who can retrieve or update information. To this end, it recognizes various directives relating to access control.

#### Traditional Access Control

Most simple access control requirements can be specified using the directives *rouser/rwuser* (for SNMPv3) or *rocommunity/rwcommunity* (for SNMPv1 or SNMPv2c).

```
rouser [-s SECMODEL] USER [noauth|auth|priv [OID | -V VIEW [CONTEXT]]]
```

```
rwuser [-s SECMODEL] USER [noauth|auth|priv [OID | -V VIEW [CONTEXT]]]
```

specify an SNMPv3 user that will be allowed read-only (GET and GETNEXT) or read-write (GET, GETNEXT and SET) access respectively.

By default, this will provide access to the full OID tree for authenticated (including encrypted) SNMPv3 requests, using the default context.

An alternative minimum security level can be specified using *noauth* (to allow unauthenticated requests), or *priv* (to enforce use of encryption). The OID field restricts access for that user to the subtree rooted at the given OID, or the named view. An optional context can also be specified, or "context\*" to denote a context prefix. If no context field is specified (or the token "\*" is used), the directive will match all possible contexts.

```
rocommunity COMMUNITY [SOURCE [OID | -V VIEW [CONTEXT]]]
```

```
rwcommunity COMMUNITY [SOURCE [OID | -V VIEW [CONTEXT]]]
```

specify an SNMPv1 or SNMPv2c community that will be allowed read-only (GET and GETNEXT) or read-write (GET, GETNEXT and SET) access respectively.

By default, this will provide access to the full OID tree for such requests, regardless of where they were sent from.

The SOURCE token can be used to restrict access to requests from the specified system(s); see *com2sec* for the full details.

The OID field restricts access for that community to the subtree rooted at the given OID, or named view. Contexts are typically less relevant to community-based SNMP versions, but the same behavior applies here.

```
rocommunity6 COMMUNITY [SOURCE [OID | -V VIEW [CONTEXT]]]
```

```
rwcommunity6 COMMUNITY [SOURCE [OID | -V VIEW [CONTEXT]]]
```

are directives relating to requests received using IPv6 (if the agent supports such transport domains). The interpretation of the SOURCE, OID, VIEW and CONTEXT tokens are exactly the same as for the IPv4 versions.

In each case, only one directive should be specified for a given SNMPv3 user, or community string. It is not appropriate to specify both *rouser* and *rwuser* directives referring to the same SNMPv3 user (or equivalent community settings). The *rwuser* directive provides all the access of *rouser* (as well as allowing SET support). The same holds true for the community-based directives.

More complex access requirements (such as access to two or more distinct OID subtrees, or different views for GET and SET requests) should use one of the other access control mechanisms. Note that if several distinct communities or SNMPv3 users need to be granted the same level of access, it would also be more efficient to use the main VACM configuration directives (see below).

## VACM Configuration

The full flexibility of the VACM is available using four configuration directives - *com2sec*, *group*, *view* and *access*. These provide direct configuration of the underlying VACM tables.

```
com2sec [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

```
com2sec6 [-Cn CONTEXT] SECNAME SOURCE COMMUNITY
```

map an SNMPv1 or SNMPv2c community string to a security name, either from a particular range of source addresses or globally ("default"). A restricted source can either be a specific hostname (or address), or a subnet; represented as IP/MASK (e.g. 10.10.10.0/255.255.255.0), or IP/BITS (e.g. 10.10.10.0/24), or the IPv6 equivalents.

The same community string can be specified in several separate directives (presumably with different source tokens), and the first source/community combination that matches the incoming request will be selected. Various source/community combinations can also map to the same security name.

If a CONTEXT is specified (using *-Cn*), the community string will be mapped to a security name in the named SNMPv3 context. Otherwise the default context ("") will be used.

`com2secunix [-Cn CONTEXT] SECNAME SOCKPATH COMMUNITY`  
is the Unix domain sockets version of *com2sec*.

`group GROUP {v1|v2c|usm|tsm|ksm} SECNAME`  
maps a security name (in the specified security model) into a named group. Several *group* directives can specify the same group name, allowing a single access setting to apply to several users and/or community strings.

Note that groups must be set up for the two community-based models separately: a *singlecom2sec* (or equivalent) directive will typically be accompanied by two *group* directives.

`view VNAME TYPE OID [MASK]`

defines a named "view" - a subset of the overall OID tree. This is most commonly a single subtree, but several *view* directives can be given with the same view name (VNAME), to build up a more complex collection of OIDs. TYPE is either *included* or *excluded*, which can again define a more complex view (e.g. by excluding certain sensitive objects from an otherwise accessible subtree).

MASK is a list of hex octets (optionally separated by '.' or ':') with the set bits indicating which sub identifiers in the view OID to match against. If not specified, this defaults to matching the OID exactly (all bits set), thus defining a simple OID subtree.

#### *Examples*

```
view iso1 included .iso 0xf0
view iso2 included .iso
view iso3 included .iso.org.dod.mgmt 0xf0
```

These directives would all define the same view, covering the whole of the *iso(1)* subtree (with the third example ignoring the sub identifiers not covered by the mask).

More usefully, the mask can be used to define a view covering a particular row (or rows) in a table, by matching against the appropriate table index value, but skipping the column sub-identifier:

```
view ifRow4 included .1.3.6.1.2.1.2.2.1.0.4 0xff:a0
```

Note that a mask longer than 8 bits must use ':' to separate the individual octets.

`access GROUP CONTEXT {any|v1|v2c|usm|tsm|ksm} LEVEL PREFIX READ WRITE NOTIFY`

maps from a group of users/communities (with a particular security model and minimum security level, and in a specific context) to one of three views, depending on the request being processed.

LEVEL is one of *noauth*, *auth*, or *priv*. PREFIX specifies how CONTEXT should be matched against the context of the incoming request, either exact or prefix.

READ, WRITE and NOTIFY specifies the view to be used for GET\*, SET and TRAP/INFORM requests (although the NOTIFY view is not currently used). For *v1* or *v2c* access, LEVEL will need to be *noauth*.

### Typed-View Configuration

The final group of directives extends the VACM approach into a more flexible mechanism, which can be applied to other access control requirements. Rather than the fixed three views of the standard VACM mechanism, this can be used to configure various different view types. As far as the main SNMP agent is concerned, the two main view types are read and write, corresponding to the READ and WRITE views of the main *access* directive.

`authcommunity TYPES COMMUNITY [SOURCE [OID | -V VIEW [CONTEXT]]]`

is an alternative to the *rocommunity/rwcommunity* directives. TYPES will usually be *read* or *read/write* respectively. The VIEW specification can either be an OID subtree (as before), or a named view (defined using the *view* directive) for greater flexibility. If this is omitted, then access will be allowed to the full OID tree. If CONTEXT is specified, access is configured within this SNMPv3 context. Otherwise the default context ("") is used.

`authuser TYPES [-s MODEL] USER [LEVEL [OID | -V VIEW [CONTEXT]]]`

is an alternative to the *rouser/rwuser* directives. The fields TYPES, OID, VIEW and CONTEXT have the same meaning as for *authcommunity*.

`authgroup TYPES [-s MODEL] GROUP [LEVEL [OID | -V VIEW [CONTEXT]]]`

is a companion to the *authuser* directive, specifying access for a particular group (defined using the *group* directive as usual). Both *authuser* and *authgroup* default to authenticated requests - LEVEL can also be specified as *noauth* or *priv* to allow unauthenticated requests, or require encryption respectively. Both *authuser* and *authgroup* directives also default to configuring access for SNMPv3/USM request. Use the *-s* flag to specify an alternative security model (using the same values as for access above).

`authaccess TYPES [-s MODEL] GROUP VIEW [LEVEL [CONTEXT]]`

also configures the access for a particular group, specifying the name and type of view to apply. The MODEL and LEVEL fields are interpreted in the same way as for *authgroup*. If CONTEXT is specified, access is configured within this SNMPv3 context (or contexts with this prefix if the CONTEXT field ends with '\*'). Otherwise the default context ("" ) is used.

`setaccess GROUP CONTEXT MODEL LEVEL PREFIX VIEW TYPES`

is a direct equivalent to the original *access* directive, typically listing the view types as *read* or *read/write* as appropriate. All other fields have the same interpretation as with *access*.

### 3.4.1.6 System Group

Most of the scalar objects in the 'system' group can be configured via the *snmpd.conf* file in this way:

`sysLocation STRING`

`sysContact STRING`

`sysName STRING`

set the system location, system contact or system name (*sysLocation.0*, *sysContact.0* and *sysName.0*) for the agent respectively. Ordinarily these objects are writable via suitably authorized SNMP SET requests. However, specifying one of these directives makes the corresponding object read-only, and attempts to SET it will result in a *notWritable* error response.

`sysServices NUMBER`

sets the value of the *sysServices.0* object. For a host system, a good value is 72 (application + end-to-end layers). If this directive is not specified, then no value will be reported for the *sysServices.0* object.

`sysDescr STRING`

`sysObjectID OID`

sets the system description or object ID for the agent. Although these MIB objects are not SNMP-writable, these directives can be used by a network administrator to configure suitable values for them.

### 3.4.1.7 Active monitoring

The usual behavior of an SNMP agent is to wait for incoming SNMP requests and respond to them; if no requests are received, an agent will typically not initiate any actions.

This section describes various directives that can configure *snmpd* to take a more active role.

#### Notification Handling

trapcommunity STRING

defines the default community string to be used when sending traps. Note that this directive must be used prior to any community-based trap destination directives that need to use it.

trapsink HOST [COMMUNITY]

trap2sink HOST [COMMUNITY]

informsink HOST [COMMUNITY]

define the address of a notification receiver that should be sent SNMPv1 TRAPs, SNMPv2cTRAP2s, or SNMPv2 INFORM notifications respectively. If COMMUNITY is not specified, the most recent *trapcommunity* string will be used.

If the transport address does not include an explicit port specification, then PORT will be used. If this is not specified, the well-known SNMP trap port (162) will be used.

If several *sink* directives are specified, multiple copies of each notification (in the appropriate formats) will be generated.

authtrappable {1|2}

determines whether to generate authentication failure traps (*disabled(2)* - the default). Ordinarily the corresponding MIB object (*snmpEnableAuthenTraps.0*) is read-write, but specifying this directive makes this object read-only, and attempts to set the value via SET requests will result in a *notWritable* error response.

v1trapaddress HOST

defines the agent address, which is inserted into SNMPv1 TRAPs. Arbitrary local IPv4address is chosen if this option is omitted. This option is useful mainly when the agent is visible from outside world by specific address only (e.g. because of network address translation or firewall).

### 3.4.2 DisMan Event MIB

The directives described in [section “Active monitoring”](#) can be used to configure where traps should be sent, but are not concerned with when to send such traps (or what traps should be generated). This is the domain of the Event MIB - developed by the Distributed Management (DisMan) working group of the IETF. The directives are described below:

`iquerySecName NAME`

`agentSecName NAME`

specifies the default SNMPv3 user name, to be used when making internal queries to retrieve any necessary information (either for evaluating the monitored expression, or building a notification payload). These internal queries always use SNMPv3, even if normal querying of the agent is done using SNMPv1 or SNMPv2c.

Note that this user must also be explicitly created (*createUser*) and given appropriate access rights (e.g. *rouser*). This directive is purely concerned with defining which user should be used - not with actually setting this user up.

`monitor [OPTIONS] NAME EXPRESSION`

defines a MIB object to monitor. If the EXPRESSION condition holds (see below), then this will trigger the corresponding event and either send a notification or apply a SET assignment (or both). Note that the event will only be triggered once, when the expression first matches. This monitor entry will not fire again until the monitored condition first becomes false, and then matches again. NAME is an administrative name for this expression, and is used for indexing the *mteTriggerTable* (and related tables). Note also that such *monitors* use an internal SNMPv3 request to retrieve the values being monitored (even if normal agent queries typically use SNMPv1 or SNMPv2c). See the *iquerySecName* token described above.

EXPRESSION

There are three types of monitor expression supported by the Event MIB: *existence*, *Boolean* and *threshold* tests.

`OID | ! OID | != OID`

defines an *existence(0)* monitor test. A bare OID specifies a *present(0)* test, which will fire when (an instance of) the monitored OID is created. An expression of the form `! OID` specifies an *absent(1)* test, which will fire when the monitored OID is detected. An expression of the form `!= OID` specifies a *changed(2)* test, which will fire whenever the monitored value(s) change. Note that there must be whitespace before the OID token.

`OID OP VALUE`

defines a *Boolean(1)* monitor test. OP should be one of the defined comparison operators (`!=`, `==`, `<`, `<=`, `>`, `>=`) and VALUE should be an integer value to compare against. Note that there must be whitespace around the OP token. A comparison such as `OID !=0` will not be handled correctly.



### OID MIN MAX [DMIN DMAX]

defines a *threshold(2)* monitor test. MIN and MAX are integer values, specifying lower and upper thresholds. If the value of the monitored OID falls below the lower threshold (MIN) or rises above the upper threshold (MAX), then the monitor entry will trigger the corresponding event.

Note that the rising threshold event will only be re-armed when the monitored value falls below the lower threshold (MIN). Similarly, the falling threshold event will be re-armed by the upper threshold (MAX).

The optional parameters DMIN and DMAX configure a pair of similar threshold tests, but working with the delta differences between successive sample values.

### OPTIONS

There are various options to control the behavior of the monitored expression. These include:

**-D**

indicates that the expression should be evaluated using delta differences between sample values (rather than the values themselves).

**-d OID**

**-di OID**

specifies a discontinuity marker for validating delta differences. A *-di* object instance will be used exactly as given. A *-d* object will have the instance sub-identifiers from the corresponding (wildcarded) expression object appended. If the *-I* flag is specified, then there is no difference between these two options.

This option also implies *-D*.

**-e EVENT**

specifies the event to be invoked when this *monitor* entry is triggered. If this option is not given, the monitor entry will generate one of the standard notifications defined in the DISMAN-EVENT-MIB.

**-I**

indicates that the monitored expression should be applied to the specified OID as a single instance. By default, the OID will be treated as a wildcarded object, and the monitor expanded to cover all matching instances.

**-i OID**

**-o OID**

define additional varbinds to be added to the notification payload when this monitor trigger fires. For a wildcarded expression, the suffix of the matched instance will be added to any OIDs specified using *-o*, while OIDs specified using *-i* will be treated as exact instances. If the *-I* flag is specified, then there is no difference between these two options.

See *strictDisman* for details of the ordering of notification payloads.

**-r FREQUENCY**

monitors the given expression every FREQUENCY seconds. By default, the expression will be evaluated every 600s (10 minutes).

**-S**

indicates that the monitor expression should not be evaluated when the agent first starts up. The first evaluation will be done once the first repeat interval has expired.

**-s**

indicates that the monitor expression should be evaluated when the agent first starts up. This is the default behavior.



Notifications triggered by this initial evaluation will be sent before the *coldStart* trap.

**-u SECNAME**

specifies a security name to use for scanning the local host, instead of the default *querySecName*. Once again, this user must be explicitly created and given suitable access rights.

**notificationEvent ENAME NOTIFICATION [-m] [-i OID | -o OID ]\***

defines a notification event named ENAME. This can be triggered from a given *monitor* entry by specifying the option *-e* ENAME (see above). NOTIFICATION should be the OID of the NOTIFICATION-TYPE definition for the notification to be generated.

If the *-m* option is given, the notification payload will include the standard varbinds as specified in the OBJECTS clause of the notification MIB definition. This option must come after the NOTIFICATION OID (and the relevant MIB file must be available and loaded by the agent). Otherwise, these varbinds must be listed explicitly (either here or in the corresponding *monitor* directive).

The *-i OID* and *-o OID* options specify additional varbinds to be appended to the notification payload, after the standard list. If the *monitor* entry that triggered this event involved a wildcarded expression, the suffix of the matched instance will be added to any OIDs specified using *-o*, while OIDs specified using *-i* will be treated as exact instances. If the *-I* flag was specified to the *monitor* directive, then there is no difference between these two options.

**setEvent ENAME [-I] OID = VALUE**

defines a set event named ENAME, assigning the (integer) VALUE to the specified OID. This can be triggered from a given *monitor* entry by specifying the option *-e* ENAME (see above).

If the *monitor* entry that triggered this event involved a wildcarded expression, the suffix of the matched instance will normally be added to the OID. If the *-I* flag was specified to either of the *monitor* or *setEvent* directives, the specified OID will be regarded as an exact single instance.

**strictDisman yes**

The definition of SNMP notifications states that the varbinds defined in the OBJECT clause should come first (in the order specified), followed by any "extra" varbinds that the notification generator feels might be useful. The most natural approach would be to associate these mandatory varbinds with the *notificationEvent* entry, and append any varbinds associated with the *monitor* entry that triggered the notification to the end of this list. This is the default behavior of the Net-SNMP Event MIB implementation.

Unfortunately, the DisMan Event MIB specifications actually state that the trigger-related varbinds should come first, followed by the event-related ones. This directive can be used to restore this strictly-correct (but inappropriate) behavior.



*strictDisMan* ordering may result in generating invalid notifications payload lists if the *notificationEvent -n* flag is used together with *monitor -o* (or *-i*) varbind options.

If no *monitor* entries specify payload varbinds, then the setting of this directive is irrelevant.

**linkUpDownNotifications yes**

will configure the Event MIB tables to monitor the *ifTable* for network interfaces being taken up or down, and triggering a *linkUp* or *linkDown* notification as appropriate.

### 3.4.3 DisMan Schedule MIB

The DisMan working group also produced a mechanism for scheduling particular actions (a specified SET assignment) at given times. This requires that the agent was built with support for the *disman/schedule* module (which is included as part of the default build configuration for the most recent distribution).

There are three ways of specifying the scheduled action:

**repeat FREQUENCY OID = VALUE**

configures a SET assignment of the (integer) VALUE to the MIB instance OID, to be run every FREQUENCY seconds.

**cron MINUTE HOUR DAY MONTH WEEKDAY OID = VALUE**

configures a SET assignment of the (integer) VALUE to the MIB instance OID, to be run at the times specified by the fields MINUTE to WEEKDAY. These follow the same pattern as the equivalent *crontab(5)* fields.



These fields should be specified as a (comma-separated) list of numeric values. Named values for the MONTH and WEEKDAY fields are not supported, and neither are value ranges. A wildcard match can be specified as '\*'

The DAY field can also accept negative values, to indicate days counting backwards from the end of the month.

at MINUTE HOUR DAY MONTH WEEKDAY OID = VALUE

configures a one-shot SET assignment, to be run at the first matching time as specified by the fields MINUTE to WEEKDAY. The interpretation of these fields is exactly the same as for the *cron* directive.

### 3.4.4 Arbitrary Extension Commands

The earliest extension mechanism was the ability to run arbitrary commands or shell scripts. Such commands do not need to be aware of SNMP operations, or conform to any particular behavior - the MIB structures are designed to accommodate any form of command output.

`exec [MIBOID] NAME PROG ARGS`

`sh [MIBOID] NAME PROG ARGS`

invoke the named PROG with arguments of ARGS. By default the exit status and first line of output from the command will be reported via the *extTable*, discarding any additional output.



Entries in this table appear in the order they are read from the configuration file. This means that adding new *exec* (or *sh*) directives and restarting the agent, may affect the indexing of other entries.

The PROG argument for *exec* directives must be a full path to a real binary, as it is executed via the *exec()* system call. To invoke a shell script, use the *sh* directive instead.

If MIBOID is specified, then the results will be rooted at this point in the OID tree, returning the exit statement as MIBOID.100.0 and the entire command output in a pseudo-table based at MIBNUM.101, with one 'row' for each line of output.



The layout of this "relocatable" form of *exec* (or *sh*) output does not strictly form a valid MIB structure. This mechanism is being deprecated - please see the *extend* directive (described below) instead.

The agent does not cache the exit status or output of the executed program.

*exec* and *sh* extensions can only be configured via the *snmpd.conf* file. They cannot be setup via SNMP SET requests.

`extend [MIBOID] NAME PROG ARGS`

works in a similar manner to the *exec* directive, but with a number of improvements. The MIB tables (*nsExtendConfigTable* etc) are indexed by the NAME token, so are unaffected by the order in which entries are read from the configuration files. There are two result tables - one (*nsExtendOutput1Table*) containing the exit status, the first line and full output (as a single string) for each *extend* entry, and the other (*nsExtendOutput2Table*) containing the complete output as a series of separate lines.

If MIBOID is specified, then the configuration and result tables will be rooted at this point in the OID tree, but are otherwise structured in exactly the same way. This means that several separate *extend* directives can specify the same MIBOID root, without conflicting.

The exit status and output is cached for each entry individually, and can be cleared (and the caching behavior configured) using the *nsCacheTable*.

Can be configured dynamically, using SNMP SET requests to the NET-SNMP-EXTEND-MIB.

### 3.4.5 Configuration example

```
## Master behavior ##
# listen for all incoming IPv4 connections on UDP port 161
# listen for all incoming IPv6 connections on UDP port 163
agentAddress  udp:161,udp6::163
# enable AgentX master on agentXSocket
master        agentx
# it is better to use TCP/IP socket for agentx communication
# if notifications are expected.
agentXSocket  tcp:127.0.0.1:705
## Access control ##

# IPv4 connections #
# grant read-write perms on all OIDs from given IP with given community
rwcommunity  snmpPriv 172.17.66.202

# grant read-only perms on system group (OID) only from all IPs with given
community
rocommunity  snmpPublic default system

# IPv6 connections #

# grant read-write perms on all OIDs from all IPs with given community
rwcommunity6 snmpIPv6

# All connections #
# grant read-only perms on all OIDs from all IPs with given user
createUser   snmpdInternalUser MD5 "password"
rouser       snmpdInternalUser

## System Information ##
sysName      AU1.BS2
sysLocation  Augsburg Limited
sysContact   admin@abg.com

## Active Monitoring ##

# send traps to the IP address with community name
trap2sink    tcp:172.17.66.202:162 publicTraps

# Event Services Configuration #
## don't forget to configure traps, as without them some part of event
services
## are pointless
# set up credentials
iquerySecName snmpdInternalUser
```

```
# This defines the traps to be sent (using notificationEvent), and explicitly
# references# the relevant notification in the corresponding monitor entry :

notificationEvent linkUpTrap linkUp ifIndex ifAdminStatus ifOperStatus
notificationEvent linkDownTrap linkDown ifIndex ifAdminStatus ifOperStatus

monitor -r 60 -e linkUpTrap "Generate linkUp" ifOperStatus != 2
monitor -r 60 -e linkDownTrap "Generate linkDown" ifOperStatus == 2

# Schedule Services Configuration #
# reload configuration once an hour, using:
repeat 3600 versionUpdateConfig.0 = 1

# reload configuration on the given our each day:
cron 10 0 * * * versionUpdateConfig.0 = 1
```



If you would like to use some other AgentX socket than `tcp:127.0.0.1:705` or `/var/agentx/master` it is better to specify it not in the `snmpd.conf`, but in the `agentx.conf` (created in the `/etc/snmp`), so all other agent will configure themselves right. As `snmpd.conf` is accessible only by master, while `agentx.conf` is accessible by master (`snmpd`) and agents.

### 3.4.6 Reconfiguring the daemon

In order to reconfigure (re-read configuration file) already running daemon one of two actions could be done:

- If you have previously configured the community/user with write permissions you can set `versionUpdateConfig.0` to `'I'` by means of `snmpset`.

*Example:*

```
snmpset -v2c -c writeComm SNMP_ADDR versionUpdateConfig.0 i 1
```

- You can call the rc script `/etc/rc2.d/S90net-snmp` under privileged user.

### 3.4.7 Configuring SNMP trap daemon `snmptrapd` (`snmptapd.conf`)

Access control checks will be applied to all incoming notifications. `snmptrapd.conf` will help us with configuring daemon's operations and how incoming traps should be processed.



If `snmptrapd` is run without a suitable configuration file (or equivalent access control settings), then such traps **will not** be processed. See the [section "Access Control"](#) for more details.

#### 3.4.7.1 `snmptrapd` behavior

`snmpTrapdAddr` [<transport-specifier>:]<transport-address>[,...]

defines a list of listening addresses, on which to receive incoming SNMP notifications. See the [section "Listening addresses in BS2000"](#) for more information.

The default behavior is to listen on UDP port 162 on all IPv4 interfaces.

`doNotRetainNotificationLogs` yes

disables support for the NOTIFICATION-LOG-MIB. Normally the `snmptrapd` program keeps a record of the traps received, which can be retrieved by querying the `nlmLogTable` and `nlmLogvariableTable` tables. This directive can be used to suppress this behavior.

`doNotLogTraps` yes

disables the logging of notifications altogether. This is useful if the `snmptrapd` application should only run `traphandle` hooks and should not log traps to any location.

`doNotFork` yes

do not fork from the calling shell.

`pidFile` PATH

defines a file in which to store the process ID of the notification receiver. By default, this ID is not saved.

#### 3.4.7.2 Access Control

It is necessary to explicitly specify who is authorized to send traps and informs to the notification receiver (and what types of processing these are allowed to trigger). This uses an extension of the VACM model, used in the main SNMP agent.

There are currently three types of processing that can be specified:

- 'log' - log the details of the notification - either in a specified file, to standard output (or `stderr`), or via `syslog` (or similar).
- 'execute' - pass the details of the trap to a specified handler program
- 'net' - forward the trap to another notification receiver.



In the following directives, `TYPES` will be a (comma-separated) list of one or more of these tokens. Most commonly, this will typically be *log,execute,net* to cover any style of processing for a particular category of notification. But it is perfectly possible (even desirable) to limit certain notification sources to selected processing only.

`authCommunity TYPES COMMUNITY [SOURCE [OID | -v VIEW ]]`

authorizes traps (and SNMPv2c INFORM requests) with the specified community to trigger the types of processing listed. By default, this will allow any notification using this community to be processed. The `SOURCE` field can be used to specify that the configuration should only apply to notifications received from particular sources

`authUser TYPES [-s MODEL] USER [LEVEL [OID | -v VIEW ]]`

authorizes SNMPv3 notifications with the specified user to trigger the types of processing listed. By default, this will accept authenticated requests. (*authNoPriv* or *authPriv*). The `LEVEL` field can be used to allow unauthenticated notifications (*noauth*), or to require encryption (*priv*), just as for the SNMP agent.

With both of these directives, the `OID` (or `-v VIEW`) field can be used to restrict this configuration to the processing of particular notifications.

`authGroup TYPES [-s MODEL] GROUP [LEVEL [OID | -v VIEW ]]`

`authAccess TYPES [-s MODEL] GROUP VIEW [LEVEL [CONTEXT]]`

`setAccess GROUP CONTEXT MODEL LEVEL PREFIX VIEW TYPES`

authorize notifications in the specified `GROUP` (configured using the *group* directive) to trigger the types of processing listed.

`createUser username (MD5|SHA) authpassphrase [DES|AES]`

See the [section “SNMPv3 authentication”](#) for a description of how to create SNMPv3 users.

`disableAuthorization yes`

will disable the above access control checks, and revert to the previous behavior of accepting all incoming notifications.

### 3.4.7.3 Notification Processing

Notifications can be forwarded on to another notification receiver, or passed to an external program for specialized processing.

`traphandle OID|default PROGRAM [ARGS ...]`

invokes the specified program (with the given arguments) whenever a notification is received that matches the `OID` token. For SNMPv2c and SNMPv3 notifications, this token will be compared against the *snmpTrapOID* value taken from the notification. For SNMPv1 traps, the generic and specific trap values and the enterprise OID will be converted into the equivalent OID (following RFC 2576).

Typically, the OID token will be the name (or numeric OID) of a NOTIFICATION-TYPE object, and the specified program will be invoked for notifications that match this OID exactly. However this token also supports a simple form of wildcard suffixing. By appending the character notification based within subtree rooted at the specified OID. For example, an OID token of 1.3.6.1.4.1\* would match any enterprise specific notification (including the specified OID itself). An OID token of 1.3.6.1.4.1.\* would work in much the same way, but would not match this exact OID - just notifications that lay strictly below this root.

Note that this syntax does not support full regular expressions or wildcards -an OID token of the form oid.\*.subids is not valid.

If the OID field is the token *default* then the program will be invoked for any notification not matching another (OID specific) *traphandle* entry.

Details of the notification are fed to the program via its standard input. Note that this will always use the SNMPv2-style notification format, with SNMPv1 traps being converted as per RFC 2576, before being passed to the program. The input format is as follows, one entry per line:

#### HOSTNAME

The name of the host that sent the notification

#### IPADDRESS

The IP address of the host that sent the notification.

#### VARBINDS

A list of variable bindings (varbinds) describing the contents of the notification, one per line. The first token on each line (up until a space) is the OID of the varbind, and the remainder of the line is its value. The formats of both of these are controlled by the *outputOption* directive (or similar configuration).

The first OID should always be *SNMPv2-MIB::sysUpTime.0*, and the second should be *SNMPv2-MIB::snmpTrapOID.0*. The remaining lines will contain the payload varbind list. For SNMPv1traps, the final OID will be *SNMPv2-MIB::snmpTrapEnterprise.0*.

#### forward OID|default DESTINATION

forwards notifications that match the specified OID to another receiver listening on DESTINATION. The interpretation of OID (and default) is the same as for the *traphandle* directive.

#### 3.4.7.4 Logging

format1 FORMAT

format2 FORMAT

specify the format used to display SNMPv1 TRAPs and SNMPv2 notifications respectively. Note that SNMPv2c and SNMPv3 both use the same SNMPv2 PDU format.

See [section “Format Specifications”](#) for the layout characters available.

ignoreAuthFailure yes

instructs the receiver to ignore *authenticationFailure* traps.



This currently only affects the logging of such notifications. *authenticationFailure* traps will still be passed to trap handler scripts, and forwarded to other notification receivers. This behavior should not be relied on, as it is likely to change in future versions.

logOption string

specifies where notifications should be logged to standard output, standard error, a specified file or via *syslog*. See the [section “Output configuration”](#) for details.

outputOption string

specifies various characteristics of how OIDs and other values should be displayed. See the [section “Output configuration”](#) for details.

### 3.4.7.5 Format Specifications

*snmptrapd* interprets format strings similarly to *printf()*. It understands the following formatting sequences:

String	Description
%%	a literal %
%a	the contents of the <i>agent-addr</i> field of the PDU (v1 TRAPs only)
%A	the hostname corresponding to the contents of the <i>agent-addr</i> field of the PDU, if available, otherwise the contents of the <i>agent-addr</i> field of the PDU(v1 TRAPs only).
%b	PDU source address (Note: this is not necessarily an IPv4 address)
%B	PDU source hostname if available, otherwise PDU source address (see note above)
%h	current hour on the local system
%H	the <i>hour</i> field from the <i>sysUpTime.0</i> varbind
%j	current minute on the local system
%J	the <i>minute</i> field from the <i>sysUpTime.0</i> varbind
%k	current second on the local system
%K	the <i>seconds</i> field from the <i>sysUpTime.0</i> varbind
%l	current day of month on the local system
%L	the <i>day of month</i> field from the <i>sysUpTime.0</i> varbind
%m	current (numeric) month on the local system
%M	the numeric <i>month</i> field from the <i>sysUpTime.0</i> varbind
%N	enterprise string
%q	trap sub-type (numeric, in decimal)
%P	security information from the PDU ( <i>community name</i> for v1/v2c, <i>user</i> and <i>context</i> for v3)
%t	decimal number of seconds since the operating system epoch
%T	the value of the <i>sysUpTime.0</i> varbind in seconds
%v	list of variable-bindings from the notification payload. These will be separated by a tab, or by a comma and a blank if the alternate form is requested
%V	specifies the variable-bindings separator. This takes a sequence of characters, up to the next % (to embed a % in the string, use \%)
%w	trap type (numeric, in decimal)
%W	trap description
%y	current year on the local system
%Y	the <i>year</i> field from the <i>sysUpTime.0</i> varbind

In addition to these values, optional fields *width* and *precision* may also be specified, just as in *printf(3)*, and a flag *value*. The following flags are supported:

- '-' (left justify)
- '0' (use leading zeros)
- '#' (use alternate form)

The "use alternate form" flag changes the behavior of various format string sequences:

- Time information will be displayed based on GMT (rather than the local time zone)
- The variable-bindings will be a comma-separated list (rather than a tab-separated one)
- The system uptime will be broken down into a human-meaningful format (rather than being a simple integer)

### *Examples*

To get a message like 14:03 TRAP3.1 from humpty.ucd.edu you could use something like this:

```
snmptrapd -P -F "%02.2h:%02.2j TRAP%w.%q from %A\n"
```

If you want the same thing but in GMT rather than local time, use

```
snmptrapd -P -F "%#02.2h:%#02.2j TRAP%w.%q from %A\n"
```

## 3.5 SNMP-AGENTS configuration

### 3.5.1 Application Monitor agent configuration

The Application Monitor agent allows monitoring of:

- user applications
- DCAM applications
- BCAM applications
- subsystems
- job variables
- log files

In addition, groups of associated statements can be managed as a unit (object).

The type and extent of the application monitoring are controlled individually via a configuration file located on BS2000 file system. The name of the configuration file is notified to the application monitor agent in the start command (in the rc file) via `-c` option. If there are errors in the configuration file, the start procedure is interrupted. If no configuration file is specified, monitoring is restricted to subsystems only.

The following table describes agent specific command line arguments:

Option	Defaults	Description
<code>-c &lt;BS2:config-file&gt;</code>	no-file; only subsystems are monitored	Apply given configuration file to start monitoring selected objects.
<code>-t &lt;sec&gt;</code>	5 sec.	Basic counter, which determines how often objects are checked: <ul style="list-style-type: none"> <li>– Subsystems checked (5 * counter) sec (default, each 25 sec)</li> <li>– Files checked (1 * counter) sec (default, each 5 sec)</li> <li>– DCAM apps checked each (60 * counter) sec (default, each 5 min)</li> </ul>

### 3.5.1.1 Statements for the configuration file

The configuration file contains information as to which applications, tasks, subsystems, job variables and log files are to be monitored. Up to 256 user applications, BCAM applications, job variables and log files can be monitored, as well as 128 DCAM applications. The user and BCAM applications and tasks to be monitored must be started with job variables. There is no limit to the number of subsystems that can be monitored.

The entries in the configuration file are generated using SDF statements. The //REMARK can be used to store comments in the configuration file. The last statement in the file must always be //END. Statements that come after the END statement are ignored.

<b>Monitoring</b>	<b>Statement</b>
Application	<a href="#">ADD-APPLICATION-RECORD</a>
DCAM application	<a href="#">ADD-DCAM-APPLICATION-RECORD</a>
Subsystem	<a href="#">ADD-SUBSYSTEM-RECORD</a>
Log file	<a href="#">ADD-LOG-FILE-RECORD</a>
Job variable	<a href="#">ADD-JV-RECORD</a>
Group of associated applications	<a href="#">DEFINE-OBJECT</a>
Monitoring intervals	<a href="#">SET-TIMER-OPTIONS</a>

## ADD-APPLICATION-RECORD

The ADD-APPLICATION-RECORD statement states the BCAM and user applications to be monitored. Applications are taken to be mean programs or tasks.

```
//ADD-APPLICATION-RECORD

APPLICATION-NAME = <composed-name_1 .. 54_with-underscore>
, VERSION = *NONE / <product-version>
, TYPE = *BCAM / *USER
, JV-NAME = <filename_1 .. 54>
, TRAP-CONDITION = A / list-poss (6) : <name_1 .. 1>
, WEIGHT= 0 / <integer 0 .. 999>
```

### APPLICATION-NAME=<composed-name\_1..54\_with-underscore>

Defines the application which the agent is to monitor.

### VERSION=\*NONE / <product-version>

Version number of the application

Default value: \*NONE

### TYPE=\*BCAM / \*USER

Type of application.

### JV-NAME = <filename\_1 .. 54>

Job variable (MONJV), which is used to monitor the application or task.

### TRAP-CONDITION=A / list-poss (6) : <name\_1 .. 1>

States for which a trap is to be generated.

### WEIGHT= 0 / <integer 0 .. 999>

Weight of the traps specific to the Application Monitor agents. When sending a (generic) trap, the Application Monitor agent supplies the specified value for the trap object *-appMonWeight* and the trap number (see [section "Notification Processing"](#)). If various weights are to be used in an application for various events, the associated ADD-APPLICATION-RECORD statement must be specified several times in the configuration file.

Default value: 0



## ADD-DCAM-APPLICATION-RECORD

The ADD-DCAM-APPLICATION-RECORD statement states the DCAM and user applications to be monitored cyclically. The monitoring interval for DCAM applications is 60 times the timer setting, i.e. 5 minutes by default. With the [//SET-TIMER-OPTIONS](#) statement, you can set the monitoring interval to any multiple of the timer setting.

A maximum of 128 DCAM applications can be monitored.

```
//ADD-DCAM-APPLICATION-RECORD

APPLICATION-NAME = <name_1 .. 8>
, HOST= *OWN / <name_1 .. 8>>
, KEEP-CONNECTION = *YES / *NO
, MSG= *NONE / <c-string> / <x-string>
, TRAP-CONDITION = list-poss (2) : *NOT-AVAILABLE / *AVAILABLE
, WEIGHT= 0 / <integer 0 .. 999>
```

### APPLICATION-NAME=<name\_1..8>

Defines the DCAM application which the agent is to monitor.

### HOST=\*OWN / <name\_1..8>

Host on which the DCAM application is running

Default value: \*OWN

### KEEP-CONNECTION=\*YES / \*NO

Defines whether the connection is to be cleared down

Default value: \*YES

### MSG= \*NONE / <c-string> / <x-string>

Connection message

### TRAP-CONDITION=\*NOT-AVAILABLE / \*AVAILABLE

Conditions under which a trap is generated.

Default value: \*NOT-AVAILABLE

### WEIGHT= 0 / <integer 0 .. 999>

Weight of the traps specific to the Application Monitor agents. When sending a (generic) trap, the Application Monitor agent supplies the specified value for the trap object *-appMon-Weight* and the trap number (see [section "Notification Processing"](#)).

Default value: 0

## ADD-SUBSYSTEM-RECORD

The ADD-SUBSYSTEM-RECORD statement defines the subsystems to be monitored. The monitoring interval for DCAM applications is 5 times the timer setting, i.e. 25 seconds by default. With the [//SET-TIMER-OPTIONS](#) statement, you can set the monitoring interval to any multiple of the timer setting.

```
//ADD-SUBSYSTEM-RECORD
```

```
NAME = <structured-name 1 .. 8> / *ALL
```

```
, VERSION = *NONE / <product-version>
```

```
, TRAP-CONDITION = *NONE / list-poss (8) : *CREATED / *NOT-CREATED / *IN-DELETE / *IN-CREATE /  
*IN-RESUME / *IN-HOLD / *NOT-RESUMED / *LOCKED
```

```
, WEIGHT= 0 / <integer 0 .. 999>
```

**NAME=<structured-name 1..8> / \*ALL**

Defines the subsystem which the agent is to monitor.

**VERSION=\*NONE / <product-version>**

Version number of the subsystem

Default value: \*NONE

**TRAP-CONDITION=\*NONE / list-poss (8) : \*CREATED / \*NOT-CREATED / \*IN-DELETE / \*IN-CREATE / \*IN-RESUME / \*IN-HOLD / \*NOT-RESUMED / \*LOCKED**

States for which a trap is to be generated.

Default value: \*NONE

*Note:*

If NAME=\*ALL is specified, you should use TRAP-CONDITION=\*NONE as otherwise performance problems may arise.

**WEIGHT= 0 / <integer 0 .. 999>**

Weight of the traps specific to the Application Monitor agents. When sending a (generic) trap, the Application Monitor agent supplies the specified value for the trap object *-appMonWeight* and the trap number (see [section "Notification Processing"](#)). If various weights are to be used in an application for various events, the associated ADD-APPLICATION-RECORD statement must be specified several times in the configuration file.

Default value: 0

## ADD-LOG-FILE-RECORD

The ADD-LOG-FILE-RECORD statement defines the log files to be monitored. By default, the Application Monitor agent sends a trap for each modification to a log file. However, it is possible to filter the traps/entries. With the [//SET-TIMER-OPTIONS](#) statement, you can set the monitoring interval to any multiple of the timer setting.

```
//ADD-LOG-FILE-RECORD
```

```
NAME = <filename_1 .. 54> / <posix-pathname>
, APPLICATION-NAME = *NONE / <composed-name_1 .. 54_with-underscore>
, MONITORING = *YES / *NO
, FORMAT = *EBCDIC / *ASCII
, PATTERN = *NONE / list-poss (8) : <c-string_1 .. 256_with-lower-case>
, WEIGHT= 0 / <integer 0 .. 999>
```

**NAME=<filename\_1 .. 54> / <posix-pathname>**

Defines the log file which the agent is to monitor.

**APPLICATION-NAME=\*NONE / <composed-name\_1 .. 54\_with-underscore>**

Name of the application.

Default value: \*NONE

**MONITORING=\*YES / \*NO**

Specifies whether the log file is to be monitored.

**FORMAT=\*EBCDIC / \*ASCII**

Format of the log file.

Default value: \*EBCDIC

**PATTERN = \*NONE / list-poss (8) : <c-string\_1 .. 256\_with-lower-case>**

Specifies one or more search patterns. If no PATTERN is specified, all entries are recorded in a log file for each trap.

The following wildcards are permitted:

?	replaces any one character
*	replaces any character string
[s]	replaces precisely one character from the s string
[c1 - c2]	replaces any character from the range c1 to c2

The "/" character (backslash) must be used as the escape character for special characters. A distinction is made between uppercase and lowercase letters.

Default value: \*NONE

**WEIGHT= 0 / <integer 0 .. 999>**

Weight of the traps specific to the Application Monitor agents. When sending a (generic) trap, the Application Monitor agent supplies the specified value for the trap object *-appMonWeight* and the trap number (see [section "Notification Processing"](#)).

Default value: 0

## ADD-JV-RECORD

The ADD-JV-RECORD statement defines the job variables to be monitored. By default, the Application Monitor agent sends each job variable modification as a trap. However, it is possible to filter the traps.

<b>//ADD-JV-RECORD</b>
<b>JV-NAME</b> = <filename_1 .. 54> , <b>APPLICATION-NAME</b> = <u>*NONE</u> / <composed-name_1 .. 54_with-underscore> , <b>PASSWORD</b> = <u>*NONE</u> / <c-string_1 .. 4 > / <x-string_1 .. 8> , <b>PATTERN</b> = <u>*NONE</u> / list-poss (8) : <c-string_1 .. 256_with-lower-case> , <b>WEIGHT</b> = <u>0</u> / <integer 0 .. 999>

### **JV-NAME = <filename\_1 .. 54>**

Defines the job variable which the agent is to monitor.

### **APPLICATION-NAME = \*NONE / <composed-name\_1 .. 54\_with-underscore>**

Name of the application.

Default value: \*NONE

### **PASSWORD = \*NONE / <c-string\_1 .. 4 > / <x-string\_1 .. 8>**

Read password of the job variables.

Default value: \*NONE

### **PATTERN = \*NONE / list-poss (8) : <c-string\_1 .. 256\_with-lower-case>**

Defines one or more search patterns. If no PATTERN is specified, all JV changes are notified per trap.

The following wildcards are permissible:

?	replaces any character
*	replaces any number of characters
[s]	replaces exactly one character in a string s
[c1 - c2]	replaces any character in the range c1 to c2

The backslash character "/" must be specified for special characters. A distinction is made between uppercase and lowercase.

Default value: \*NONE

**WEIGHT= 0 / <integer 0 .. 999>**

Weight of the traps specific to the Application Monitor agents. When sending a (generic) trap, the Application Monitor agent supplies the specified value for the trap object *-appMonWeight* and the trap number (see [section "Notification Processing"](#)).

Default value: 0

## DEFINE-OBJECT

Logically associated components in a process (applications, logfiles, subsystems and job variables) can be grouped together using the statement DEFINE-OBJECT. All elements stated in the DEFINE-OBJECT statement must also be defined in the configuration file with the corresponding ADD... statement.

If the specifications made for an element of the object in the DEFINE-OBJECT for ICON and ACKNOWLEDGE contradict the corresponding specifications in the ADD... statement, the specifications made in the DEFINE-OBJECT statement apply.

```
//DEFINE-OBJECT
```

```
OBJECT-NAME = <composed-name_1 .. 8_with-underscore>
, BCAM-APPLICATION = *NONE / list-poss(5): <composed_name_1 .. 54_with-underscore>
, USER-APPLICATION = *NONE / list-poss(5): <composed_name_1 .. 54_with-underscore>
, DCAM-APPLICATION = *NONE / list-poss(5): <name_1 .. 8>
, LOG-FILE = *NONE / list-poss(5): <filename_1 .. 54> / <posix-pathname>
, SUBSYSTEM = *NONE / list-poss(5): <structured-name_1 .. 8>
, JV = *NONE / list-poss(10): <filename_1 .. 54>
, MONITORING-TIME = *ALWAYS / *INTERVAL (...)
  *INTERVAL (...)
    , START-TIME = hh:mm
    , STOP-TIME = hh:mm
, EXCEPT-DAYS = *NONE / list-poss(6): MON / TUE / WED / THU / FRI / SAT / SUN
```

**OBJECT-NAME = <composed-name\_1 .. 8\_with-underscore>**

Name of the object

**BCAM-APPLICATION = \*NONE / list-poss(5): <composed\_name\_1 .. 54\_with-underscore>**

BCAM applications that belong to this object

Default value: \*NONE

**USER-APPLICATION = \*NONE / list-poss(5): <composed\_name\_1 .. 54\_with-underscore>**

User applications that belong to this object

Default value: \*NONE

**DCAM-APPLICATION = \*NONE / list-poss(5): <name\_1 .. 8>**

DCAM applications that belong to this object

Default value: \*NONE

**LOG-FILE = \*NONE / list-poss(5): <filename\_1 .. 54> / <posix-pathname>**

Log files that belong to this object

Default value: NONE

**SUBSYSTEM = \*NONE / list-poss(5): <structured-name\_1 .. 8>**

Subsystems that belong to this object

Default value: \*NONE

**JV = \*NONE / list-poss(10): <filename\_1 .. 54>**

Job variables that belong to this object

**MONITORING-TIME = \*ALWAYS / \*INTERVAL (...)**

Specifies the monitoring time

Default value: \*ALWAYS

**\*INTERVAL ( ...)**

Defines the monitoring interval. If STOP-TIME is greater than START-TIME, the hours after midnight are counted to the previous day when checking the EXCEPT-DAYS.

*Example:*

The monitoring time ranges from 20:00 to 3.00 hrs, except from Saturday and Sunday. Monitoring therefore stops on Saturday at 3:00 in the morning and starts again on Monday at 20:00 in the evening.

**START-TIME = HH:MM**

Time when the object should be monitored

**STOP-TIME = HH:MM**

Time up to which the object should be monitored

**EXCEPT-DAYS = \*NONE / list-poss(6): MON / TUE / WED / THU / FRI / SAT / SUN**

Weekdays on which the object is not to be monitored

Default value: \*NONE

*Example: Monitoring a MAREN systems*

A MAREN system consists of the following components:

- Subsystem MAREN
- Control program MARENCP
- Automatic assignment of free tape MARENUCP

In addition, each VSN reserved by the automatic tape assignment function is automatically stored in the job variable TAPE.FILE.MAREN.

The following definition of a "MAREN" object combines these components:

```
//DEFINE-OBJECT OBJECT-NAME = MAREN
//, USER-APPLICATION = (MARENCP, MARENUCP)
//, SUBSYSTEM = MAREN
//, JV = TAPE.FILE.MAREN
```



## SET-TIMER-OPTIONS

The Application Monitor agent uses a timer. You specify the value for the timer when starting the Application Monitor agent. The statement SET-TIMER-OPTIONS defines the monitoring interval (polling factor). The monitoring interval defines the number of timer cycles which are to elapse before the next check is to be carried out.

```
//SET-TIMER-OPTIONS  
  
FILES = 1 / <integer>  
, SUBSYSTEMS = 5 / <integer>  
, DCAM-APPLICATIONS = 60 / <integer>
```

### FILES = 1 / <integer>

Defines the polling factor for files.

Default value: 1

### SUBSYSTEMS = 5 / <integer>

Defines the polling factor for subsystems.

Default value: 5

### DCAM-APPLICATIONS = 60 / <integer>

Defines the polling factor for DCAM applications.

Default value: 60

### 3.5.1.2 Change in the configuration file during the current session

Changes to the current configuration file during a session can be made by the Application Monitor by setting the *appMonConfFile0* object.

#### *Example*

```
snmpset -v2c -cpublic SNMP_ADDR appMonConfFile.0 s "APPMON.CONF"
```

If there are syntax errors in *appMonConfFile*, the original configuration is retained.

### 3.5.2 Configuring the Console Monitor agent

The Console Monitor has access to the console commands of \$CONSOLE via UCON. The following preparation is required to enable the Console Monitor to access the BS2000 console:

- Create the operator ID <operator-id>
- Granting the access rights for the operator user ID

*Create the operator ID <operator-id>*

```
/ADD-USER USER-ID=<operator-id>, -
          PROTECTION-ATTRIBUTE=*PAR(LOGON-PASSWORD=<pass>), -
          ACCOUNT-ATTRIBUTES=*PAR(ACCOUNT=<account-no>)
```

The logon attributes defined here must be specified in the rc file when starting the Console Monitor (see [Console Monitor agent](#)).

*Granting the access rights for the operator user ID*

For operating with SECOS, access rights must additionally be granted for the operator ID to \$CONSOLE:

```
/MOD-LOGON-PROTECTION USER-IDENTIFICATION=<operator-id>, -
                      OPERATOR-ACCESS-PROG=*YES(PASSWORD-CHECK=*YES)
```

The class 2 system parameter NBBAPRIV must be set to the default value N.

The following tables describe agent specific command line arguments:

Mandatory Options	Description
– -o <operid>	– Specify Operator's ID
– -y <op-role1> [, <op-role2>, ....., <op-role10>]	– Specify Operator's role(-s); up to 10
or	
– -a <auth-file>	– Load credentials specified in file

Optional arguments	Description
-k <password>	Specify password to access \$CONSOLE
-c <msg-filter>	Load message filter
-n <negative-msg-filter>	Load negative message filter
-A <auth-file>	Outputs a prompt with which an authentication file can be generated. This file remains encrypted with AES.

Optional arguments	Description
-s <auth-file>	Outputs the operator IDs and role(s) defined in the authentication file. If a password has been defined, *SET is output, otherwise *NONE.

### Defining message filters

The Console Monitor agent uses two filter options for selecting messages:

- positive message filter  
selects messages to be sent to the management station
- negative message filter  
selects messages not to be sent to the management station

#### 3.5.2.1 Positive message filter

The following two filter options are available for selection of messages to be sent to the management station:

- Routing code (assigned to each console message)
- Message key (uniquely identifies each message)

##### *Routing code selection criterion*

Each message is assigned a specific routing code. Operator roles contain the routing codes of the messages to be sent to the management station. The operator roles are specified when starting the Console Monitor (see [section “Agents-specific options for starting agents manually”](#)). The following statements show you how operator roles are created and assigned to the operator ID. The SECURITY ADMINISTRATION privilege, which the user ID SYSPRIV has as default, is required for issuing the following statements.

Create the operator role:

```
/CREATE-OPERATOR-ROLE OP-ROLE=<op-role-name>, -
                        ROUTING-CODES=.....
```

Assign the operator role to the operator ID:

```
/MODIFY-OPERATOR-ATTR USER-ID=<operator-id>, -
                        ADD-OPERATOR-ROLE=(<op-role-name1>,...,<op-role-namex>)
```

The operator ID must additionally be assigned the OPERATING privilege if SECOS is used:

```
/SET-PRIVILEGE PRIV=OPERATING,USER-ID=<operator-id>
```

*Message code selection criterion*

The codes of messages to be sent to the management station are stored in the positive message filter file. The statements

- *msgid*
- *QUESTION*
- *TYPIO*

provide three filter options. The name of the message filter file is made known to the Console Monitor when it is started via the *-c* command argument or it can be entered in the MIB *consMonMsgFilter* object during a session.

If no message filter file is specified when the Console Monitor is started, all messages are output for which the routing code is specified in the operator role.

If the message filter file contains no key, or no valid key, no traps are sent to the management station.

The following name conventions apply to the message filter file:

/BS2/<file>	BS2000 file
[:<catid>:]\$<userid>.<file>	BS2000 file
*POSIX(<file>)	POSIX file
/<path>/<file>	POSIX file
<file>	The deciding factor in this case is the environment in which the agent was started.

### 3.5.2.2 Structure of the positive message filter

#### msgid statement

```
<msgid [wgt] [SOURCE=src] [DEVICE=dev]
    [PATTERN=/pat1[/..patx]] [[ACKNOWLEDGE=YES]]>
```

#### msgid

Specifies a message code.

The following wildcards are permitted for message code entries:

?	replaces any character
*	replaces any number of characters
[s]	replaces exactly one character in a string s
[c1 - c2]	replaces any character in the range c1 to c2

The backslash character "/" must be specified for special characters. A distinction is made between uppercase and lowercase.

**wgt**

Specifies a message weight. A weight can be assigned to the message codes. The weight is prefixed to the actual message in the trap string. This allows the user to set the importance of messages himself and transmit this to the management station. The message code is assigned the value 0 as default if no weight is specified. The entry is expected as an integer in the range 0 - 999.

**src**

Specifies a source name. The source is supplied with BS2-<source> in the trap string. The default value *BS2Console* is used if no value is specified. You can set an alarm to a specific object in the network map with this entry. The entry is alphanumeric in the range 1 - 12.

**pat**

Specifies one or more search patterns (pattern).

?	replaces any character
*	replaces any number of characters
[s]	replaces exactly one character in a string s
[c1 - c2]	replaces any character in the range c1 to c2

The character "\" (backslash) must be specified to invalidate special characters. A distinction is made between upper case and lower case.

**dev**

If DEVICE is specified, the Console Monitor agent sends this trap with the DEVICE entry as Community.

ACKNOWLEDGE=YES

If you specify ACKNOWLEDGE=YES, the agent is informed that this trap must be acknowledged.

**QUESTION statement**

Question filters all messages that contain a question, i.e. expect an answer. If a question is encountered, the Console Monitor first checks whether a pattern of QUESTION entries matches. If not, the MSGID entries or the TYPIO entries are searched for the relevant message type.

```
<QUESTION [wgt] [SOURCE=src] [DEVICE=dev] [PATTERN=/pat1[/..patx]]
[ACKNOWLEDGE=YES]>
```

**QUESTION**

Message key of a console query

wgt

see above

src

see above

dev

see above

pat

see above

**ACKNOWLEDGE=YES**

see above

*Example:*

<QUESTION PATTERN=[0-9]*>	Selection of all questions that start with a digit.
---------------------------	---

**TYPIO statement**

TYPE I/O messages are a special case. These include, for example, messages sent to the BS2000 console with /SEND-MSG. Their reception is also controlled via the message filter file. The entry for a TYPE I/O message is as follows:

```
<TYPIO [wgt] [SOURCE=src] [DEVICE=dev] [PATTERN=/pat1[/.patx]] [ACKNOWLEDGE=YES]>
```

wgt

see above

src

see above

dev

see above

pat

see above

**ACKNOWLEDGE=YES**

see above

*Example:*

<pre>&lt;TYPIO PATTERN=/*abc*/xyz&gt; &lt;TYPIO PATTERN=/Hallo*&gt; &lt;TYPIO PATTERN=/?\?*&gt;</pre>	<p>All TYPE I/O messages that contain the string "abc", are made up only of "xyz", start with "Hello" or have a question mark as their second character, are sent to the management station as a trap.</p>
---	--

### 3.5.2.3 Negative message filter

A negative message filter is also provided as of Console Monitor agent. The message code of the console messages, which are not to be forwarded to the management station are stored in the negative message filter file. Questions cannot be suppressed. The MIB object *consMonNegMsgFilter* refers to the name of the negative message filter file. The name of the negative message filter file is defined with the *-n* option when the Console Monitor is started. This definition can only be modified when the Console Monitor is started, not during a session.

The length of the entry must not exceed 179 characters.

```
<msgid> [PATTERN=/pat1[/../patx]]> [<msgid [PATTERN=/pat1[/../patx]]>] ...
```

For the description of *msgid* and *pat* see "[Structure of the positive message filter](#)".

### 3.5.2.4 Modifying the configuration file during operation

It is possible to modify the current message filter file during operation using the Console Monitor application by setting the *consMonMsgfilter* object.

If the *consMonMsgFilter* file contains syntax errors, processing continues with the original message filter file.

*Example: filtering console messages*

The message EXC0858 should only be sent to the management platform if it contains neither the string "CLAQ" nor the string "TEST". The trap should be sent with the trap number 99 and have "Hardware" entered as its source.

This is done as follows:

1. In the positive message filter, enter: <EXC0858 99 SOURCE=Hardware>
2. In the negative message filter, enter: <EXC0858 PATTERN = \*CLAQ\* / \*TEST\*>



### 3.5.3 Configuring the Storage agent

You can use the storage management agent to monitor disks and pubsets. For this pass the configuration file via the command line on the startup (configure it in the rc file):

```
"-c <BS2:config-file>".
```

It is possible to modify the configuration file during operation by setting the *storMgmtGlobalDataInputFile.0* object. If the configuration file contains syntax errors, processing continues with the original message filter file.

The configuration is performed using an input file:

- For monitoring the saturation level of individual public volumes (pubsets), the relevant PVS must be specified in the input file of the agent. This is done using the ADD-PUBSET-RECORD statement.
- To monitor the state of the selected ROBAR application, use the statement ADD-ROBAR-RECORD.
- To monitor the reconfiguration state of the individual disks, the relevant disks must be specified in the input file of the agent. This is done using the ADD-DISK-RECORD statement.
- The //REMARK statement can be used to store comments in the configuration file.
- The last statement in the configuration file should be the //END statement. Any statements that appear after the //END statement are ignored.
- A maximum of 128 pubsets and/or disks can be monitored.

### ADD-PUBSET-RECORD - adding a pubset to be monitored

```
//ADD-PUBSET-RECORD
```

```
PUBSET= <cat_id 1..4>
```

```
, CHECK=SATURATION-LEVEL
```

```
, TRAP-COMMUNITY= *STORAGE / *PUBSET-NAME / <c-string 1..64>
```

**PUBSET=<cat\_id 1..4>**

CAT-ID of the pubset to be monitored.

**CHECK=~~SATURATION-LEVEL~~**

Object to be monitored; currently on possible to specify the SATURATION-LEVEL (Default value).

**TRAP-COMMUNITY=\*STORAGE / \*PUBSET-NAME / <c-string 1..64>**

Community string with which the trap is sent.

If \*PUBSET is defined, the <cat-id> is used as the Community Name.

If <c-string 1..64> is specified, this string is sent as the Community Name.

Default value: \*STORAGE

**ADD-DISK-RECORD - adding a disk to be monitored**

```
//ADD-DISK-RECORD
```

```
DISK-MN =<alphanum-name 1..4>
```

```
, CHECK=RECONFIGURATION-STATE
```

```
, TRAP-COMMUNITY= *STORAGE / *DISK-MN / <c-string 1..64>
```

**DISK-MN=<alphanum-name 1 ..4>**

Mnemonic name for the device to be monitored

**CHECK=RECONFIGURATION-STATE**

Object to be monitored; currently on possible to specify the RECONFIGURATION-STATE (Default value).

**TRAP-COMMUNITY=\*STORAGE / \*DISK-MN / <c-string 1..64>**

Community string with which the trap is sent.

If \*DISK-MN is defined, the name defined for DISK-MN is used as the Community Name.

If <c-string 1..64> is specified, this string is sent as the Community Name.

Default value: \*STORAGE0

### ADD-ROBAR-RECORD - adding a ROBAR application to be monitored

```
//ADD-ROBAR-RECORD
```

```
LOCATION=< composed-name_1..8_with-underscore>
```

```
, VERSION=*NONE / <product-version mandatory-man-corr> / <product-version without-man-corr>
```

```
, JV-NAME= <filename_1..54>
```

```
, TRAP-CONDITION = A / list-poss (6) : <name_1 .. 1>
```

**LOCATION=< composed-name\_1..8\_with-underscore>**

Location of the ROBAR application.

**VERSION=\*NONE / <product-version mandatory-man-corr> / <product-version without-man-corr>**

Version of the application to be monitored. If a version number is specified, the format specified here must be identical to the format used when the subsystem was defined (release and correction status mandatory or not allowed).

**JV-NAME= <filename\_1..54>**

Job variable (MONJV), which is used to monitor this application.

**TRAP-CONDITION = A / list-poss (6) : <name\_1 .. 1>**

States of a MONJV for which a trap is sent. The value A means all states.

### 3.5.4 Configuring the openUTM agent

The following section describes the activities required for putting the openUTM agent into operation.

#### 3.5.4.1 Preparation

The agent communicates with a UTM application via UPIC(BS2000). UPIC requires a side information file (*upicfile*) to link the agent to the UTM application. This file must be named *upicfile* and be cataloged in BS2000. Normally, there are four parts to the entry in the *upicfile*:

- An identifier; in this case, HD as the identifier for a link between UPIC(BS2000) and UTM(BS2000).
- The communication partner of the application, which should be set to SNMP4UTM for the currently selected UTM application.
- The partner name presented by the main BCAM name of the application. It is recommended to specify application's host machine, separated from the name by ".".
- The transaction code; this entry is not necessary in this case because as the agent specifies the transaction code with the *Set\_TP\_Name* call.

The file *upicfile* can be edited under BS2000. The end-of-line character is represented in BS2000 by a semicolon (";") as there is no <newline> character in BS2000 (see example). This means that if an edited line contains a semicolon, UPIC interprets this as the end of the line and puts the remainder on the next line (up to the next semicolon). This also applies to comment lines.

#### *Example*

##### Side Information file:

```
*symbolic destination names for (BS2000) application ZENTRBS2;  
;*application is running on BENGINE;  
HDSNMP4UTM ZENTRBS2.BENGINE;
```

The UTM agent reports to the UPIC communication system with the local name SNMPUPIC. The name SNMP4UTM should be defined as the communication partner of the application with the KDCDEF PTERM or TPOOL statement.

Since the UTM agent requires the appropriate authorization to issue UTM administration commands, a UTM user ID must be defined with STATUS=ADMIN or PERMIT=ADMIN and passed to the UTM agent with help of respective job variables, see [section "Runtime environment"](#). If no JVs will be created, agent will use default credentials from SNMP-AGENTS.

openUTM uses the following TACs for monitoring with SNMP:

- KDCWADMI for reading information
- KDCLPAP, KDCLTAC, KDCSHUT, KDCPOOL, KDCSWITCH, KDCPTerm, KDCTCL, KDCLTAC, KDCUSER for setting and modifying object parameters.

Please ensure that these TACs are generated in the UTM application with the required administration authorization.

### 3.5.4.2 Configuring the openUTM agent for monitoring several UTM applications

To monitor several openUTM applications state, a configuration file is required in which every monitored openUTM application must be specified.

The configuration file can be passed to the agent on the startup (in the rc file) via the command line argument:

```
-c <BS2:config-file>.
```

It is possible to modify the configuration file during operation by setting the *utmGlobalConfFile.0* object. If the configuration file contains syntax errors, processing continues with the original message filter file.

Entries in the configuration file are generated with the SDF statement `//ADD-APPLICATION-RECORD`. With the statement `//REMARK`, comments can be stored in the configuration file. The file must be terminated with the statement `//END`.

**ADD-APPLICATION-RECORD**

The statement //ADD-APPLICATION-RECORD identifies the UTM applications to be monitored.

```
//ADD-APPLICATION-RECORD
```

```
APPLICATION-NAME = <structured-name 1 .. 8>
```

```
, FILEBASE= *SAME / <full-filename_1 .. 54>
```

```
, USER-ID= TSOS / <name_1 .. 8>
```

```
, TRAP-CONDITION = list-poss (3): *ABNORMAL-TERMINATED / *NORMAL-TERMINATED / *RUNNING
```

**APPLICATION-NAME=<structured-name 1 .. 8>**

Defines the UTM application which the agent is to monitor.

**FILEBASE= \*SAME / <full-filename\_1 .. 54>**

Basic name of the A-parts of the KDCFILE.

\*SAME is the default value.

**USER-ID=TSOS / <name 1 .. 8>**

ID under which the UTM application is started.

TSOS is the default value.

**TRAP-CONDITION= list-poss (3): \*ABNORMAL-TERMINATED /  
\*NORMAL-TERMINATED / \*RUNNING**

Conditions under which a trap is to be generated.

\*ABNORMAL-TERMINATED is the default value.

### 3.5.4.3 Runtime environment

The UPIC program is controlled via job variables in the BS2000 system. UPIC evaluates the following job variables:

Job variable	Link name	Meaning
UPICPATH	*UPICPAT	The job variable UPICPATH defines the file directory under which the side information file is stored. If the job variable is not set, the current file directory is used for the search. If the agent is started under POSIX, the job variable UPICPATH must be supplied the value "BS2/\$<userid>2, since UPIC would otherwise try to open the <i>upicfile</i> in the POSIX file system.
UPICFILE	*UPICFIL	Specifies the right-hand part of the name of the side information file. If this variable is not set, the file name is set to <i>upicfile</i> . The complete file name is composed of UPICPAT.UPICFIL. If neither UPICPAT nor UPICFIL is set, the file name is "\$progid.UPICFILE".
UPIC4SNMP.USER	---	UTM user ID used by the openUTM agent to issue administration commands. If the job variable is not set, USER=SNMPADM is used.
UPIC4SNMP.PASS	---	Password for the UTM user ID. If the job variable is not set, USER=SNMPUPIC is used.
UPICTRACE	*UPICTRA	The job variable UPITRACE controls the trace generation (see <a href="#">section "Diagnostic documents"</a> ).
UPICLOG	*UPICLOG	The job variable UPICLOG defines the name of the logging file. If this definition is missing, the name is "##.USR.TMP.UPICL<tsn>".

Note that the assignment is lost after LOGOFF.



### 3.5.4.4 Diagnostic documents

Besides the trace file, of the openUTM agent, there are other files which may be helpful in the event of an error:

- UPIC-Trace file
- UPIC-Logging file
- SYSLOG file

#### UPIC-Trace file

The carries system UPIC enables trace information to be generated for all interface calls. You can control this procedure by setting the job variable *UPICTRACE*. The call *Enable\_UTM\_UPIC* interprets the contents of the job variable. If the job variable is set, the parameters and the user data are logged project-specific in a file up to a size of 128 bytes.

#### *Activating the UPIC trace*

The UPIC-Trace is activated as follows:

```
/SET-JV-LINK LINK-NAME=*UPICTRA,JV-NAME=UPICTRACE
/MODIFY-JV UPICTRACE,VALUE='-S[X] [-R wrap] [-Dprefix]'
```

Meaning:

-S	Detailed logging of call, the associated arguments and user data up to a maximum length of 128 bytes (mandatory data)
-SX	Additional internal information at the interface to the transport system are logged.
-R <i>wrap</i>	The decimal number specified by <i>wrap</i> defines the maximum size of the temporary trace file. Default value: 128.
-D <i>prefix</i>	The trace files are created with the following names: – <i>prefix</i> .UPICT<tsn> – <i>prefix</i> .UPICU<tsn> If <i>prefix</i> is not specified, "##.USR.TMP" is used as prefix.

#### *Deactivating the UPIC trace*

The UPIC-Trace is deactivated using the following two commands:

```
/DELETE-JV UPICTRACE
/MODIFY-JV UPICTRACE,VALUE=''
```

**UPIC logging file**

If the *openUTM* application terminates a conversation normally, an *openUTM* error message is written to the UPIC logging file. The UPIC logging file is only opened to write the error message (mode *append*) and then closed again.

**SYSLOG file**

When an application is started, *openUTM* creates an application-specific log file SYSLOG. This file logs events which occur during the application runtime in the form of *openUTM*-messages.

### 3.5.5 Configuring the openSM2 agent

The following section describes the activities required for putting the openSM2 agent into functional state. Without those steps openSM2 agent is useless and returns no-data(-1) state for most of the objects.

Prerequisites:

- SM2 subsystem is created.
- SM2 measurement is configured and enabled. This could be done by executing the following statements in BS2000:

```
/EXEC $SM2
*call-admin-part
  //set-periodic-task-parameter log-tasks=*none
  //start-measurement-program per
  //start-measurement-program utm
  //call-eval-part
*end
```



If openUTM performance monitoring is expected, openSM2 monitoring has to be activated for the respective UTM applications, e.g. by using the UTM administration command `KDCAPPL SM2=ON`.

### 3.5.6 Configuring the HSMS agent

In order to put agent into the working state it is mandatory to pass the name of the system's `SYS-LIB.HSMS`. It should be done via the following command line argument:

```
'-1 <SYSLIB.HSMS>'
```

*Example*

```
hmsAgent -1 \${SYSHSMS}.SYSLIB.HSMS.110 &
```

### 3.5.7 TCP-IP-AP configuration

This section describes prerequisites for starting the FTP daemon via *ftpAgent*.

First, the SYSENT.TCP-IP-AP.053.FTPD start procedure must exist. This file contains startup parameters for the FTP daemon.

Starting new the FTP daemon with given port and FTAC level is possible using *snmpset* only if two options are set up inside the SYSDAT.TCP-IP-AP.053.FTPD.OPT:

```
FTACLevel | -B
```

```
childName | -C
```

After that you can start the daemon with default *FTACLevel (0)* like this:

```
snmpset [OPTIONS] ftpServerPort.portNumber i portNumber
```

where *portNumber* is a new server port for controlling the connection to the FTP clients.

For more information, please refer to the latest "interNet Services - System Administrator Guide".

---

## 4 Operations

The delivery unit NET-SNMP contains the SNMP daemon (*snmpd*), the SNMP trap daemon, the SNMP client tools and the functionality of both Event and Scheduling services.

With SNMP-AGENTS, a set of agents for system and administration tasks is supplied.

In addition, the products BCAM, SESAM/SQL and TCP-IP-AP provide product-specific agents, which supplement the functionality of SNMP-AGENTS.

This chapter describes startup and shutdown of the separate components in BS2000/OSD as well as commands used to retrieve information. The final section provides information on the procedure in the event of errors.



### Important!

- None of the agents are "daemonized" by default and it is recommended to start them in the background by adding "&" at the end of statement.
- It is **not** recommended to start agents manually. The best practice is to use rc scripts to start/stop the SNMP products.

## 4.1 rc scripts

rc scripts are installed for all SNMP related products, which permit an automatic start of the agent on startup of POSIX or an automatic stop when POSIX is terminated. By default, all of the starting rc files are delivered commented out. In order to enable the automatic startup of NET-SNMP or any of its agents, their respective rc files should be modified to match your configuration.



Some agents need mandatory configuration in the rc files, like passing configuration files and/or credentials. Keep in mind the limitations described in [section "Parallel operation of SNMP V6.x and NET-SNMP"](#).

### Example (SNMP daemon)

In order to start *snmpd* automatically, you have to remove the comment in the following line of the start script */etc/rc2.d/S90net-snmp*:

```
#!/opt/net-snmp/snmpd $_OPTIONS &
```

### Example for Console Agent

From original rc file:

```
_OPT="-LS0-6d -p /var/run/consoleAgent.pid"
# IMPORTANT! Don't forget to add mandatory options:
# -o Operator's ID; -y Operator's Role(-s); -k password;
# -a authentication file's path (use this option for auto-start)
# To create authentication file run agent with -A and follow the prompt.
# uncomment following line to start subagent consoleAgent
# /opt/snmp-agents/consoleAgent $_OPT &
```

In order to start the Console Agent automatically, you have to do the following:

- Change variable `_OPT` by adding to it mandatory options/credentials, e.g.:

```
_OPT="-LS0-6d -p /var/run/consoleAgent.pid -o tsos -y sysadm -k 12345678"
```

- Uncomment line which calls the agent itself.

#### Note

Using plain-text passwords is not safe. The Console Agent facilitates the usage of authentication files, which are AES-encrypted. In order to create such file proceed as follows:

- Call `"consoleAgent -A"`
- Follow the prompt
- Edit rc file by passing argument `-a` with absolute PATH to the created *auth-file*.

## 4.2 NET-SNMP daemons and SNMP tools

### 4.2.1 SNMP daemon `snmpd`

Before the `snmpd` is started for the first time, the `/etc/snmp/snmpd.conf` file in the BS2000 system must be matched to the configuration used (see [section “System Group”](#)).

In order to start the SNMP daemon, the BS2000 user ID must possess the POSIX User ID 0 (SYSROOT).



NET-SNMP comprises the functionality of MIB-II, Event Services and Scheduling Services. This functionality can be used as soon `snmpd` has been started.

#### **snmpd configuration examples**

##### *Examples*

```
1. snmpd -LS0-4d -Lf /var/adm/snmpd.log -p /var/run/snmpd.pid -a &
```

`-LS0-4d`

Logging to the syslog with maximum log level LOG\_WARNING.

`-Lf /var/adm/snmpd.log`

Log to the specified file.

`-p /var/run/snmpd.pid`

Saves daemon's PID to file.

`-a`

Log the source addresses of incoming requests.

`-q`

Print information in a format which is simple to parse.

This `snmpd` will read standard `snmpd.conf` from `/etc/snmp` folder.

```
2. snmpd -C -c /etc/snmp/example.conf &
```

`-C`

Don't use default configuration file.

`-c`

Load given configuration file.

3. `snmpd -C --rwcommunity=public --master=agentx -x tcp:705 tcp:1111 &`
  - C  
Don't use default configuration file.
  - rwcommunity=public  
Set read/write community "public".
  - master=agentx  
Enable AgentX protocol.
  - x tcp:705  
Listen for AgentX connections on address.
  - tcp:1111  
Listen for SNMP's requests on address.

## 4.2.2 SNMP trap daemon `snmptrapd`

By default, `snmptrapd` listens for incoming SNMP requests in UDP port 162 for all IPv4 addresses. You can modify the setting by changing its configuration file (by default, `snmptrapd.conf`) or by passing command line arguments. For more information on this, please refer to [section "Configuring SNMP trap daemon `snmptrapd` \(`snmptrapd.conf`\)"](#).



`snmptrapd` should be started prior to the `snmpd`.

### Snmptrapd configuration examples

1. `snmptrapd -C -c ./trap.conf -Lo &`
  - C  
Don't use default configuration file.
  - c  
Load given configuration file.
  - Lo  
Log to the stdout.
2. `snmptrapd -C -Lo --authcommunity="log public" udp:1162 &`
  - authcommunity="log public"  
Log all traps with trapcommunity "public".
  - udp:1162  
Listen for trap notifications from all IPs on given socket.



## 4.2.3 SNMP tools `snmpwalk`, `snmpget` and `snmpset`

### `snmpwalk`

`snmpwalk` allows you to retrieve subtrees.

For more information on this subject, please consult the `snmpwalk` instructions page, available on the official Net-SNMP open-source repositories.

#### *Example*

The following command will retrieve all of the variables under `system`:

```
snmpwalk -Os -c public -v 1 zeus system
```

`-Os`

(parsing option) Print only last symbolic element of OID (OID = object identifier).

`-c public`

where `public` is the community name.

`-v 1`

Will use 1st version of the SNMP protocol (community based).

`zeus`

Name of the target system.

`system`

OID name of the network entity.

### `snmpget` and `snmpset`

`snmpget` is used to query for information on a network entity.

`snmpset` is used to set information on a network entity.

#### *Examples*

1. `snmpget -c public zeus system.sysDescr.0`

Will return value of `sysDescr.0` OID if available.

2. `snmpset -c private -v 1 test-hub system.sysContact.0 s dpz@noc.rutgers.edu`

Will set `sysContact.0` OID to the string value `dpz@noc.rutgers.edu`.

`test-hub`

Name of the target system.

system.sysContact.0

OID of the object for which the value is to be set.

s

type of variable that is to be set (here: string)

dpz@noc.rutgers.edu

New value.

## 4.3 Starting and stopping the agents of SNMP-AGENT



It is **not** recommended to start agents manually. The best practice is to use rc scripts to perform start/stop of the SNMP products.

The SNMP-AGENTS V1.0 delivery unit comprises the following agents:

- `appMonAgent` - Application Monitor Agent for monitoring subsystems, BCAM and user applications as well as job variables and logging files
- `consoleAgent` - Console Monitor Agent for monitoring the console
- `hostAgent` - Host Resources Agent to get information about the system, devices, file systems, and the installed software.
- `hsmsAgent` - HSMS Agent for monitoring the storage management system HSMS
- `openFTAgent` - openFT Agent for monitoring the openFT file transfers
- `openSM2Agent` - openSM2 Agent for monitoring the performance
- `utmAgent` - openUTM Agent for monitoring UTM applications
- `spoolAgent` - Spool Agent for monitoring SPOOL and RSO devices
- `storageAgent` - Storage Agent for monitoring disks and pubsets

The agents can be started/stopped using rc scripts, see [section “rc scripts”](#). This is the recommended practice.

In addition, it is possible to start the agents individually in POSIX using its program name, see also [section “Agents-specific options for starting agents manually”](#).

### Prerequisites

Prerequisites for starting the agents manually are:

- an operational LAN connection between BS2000 system and management platform
- a started POSIX subsystem
- an installed SNMPAGT subsystem
- the SNMP daemon `snmpd` is running.

The SYSROOT privilege is required to start the agents:

### 4.3.1 Agents-specific options for starting agents manually

This section describes only agent-specific options. All other agents could be started without specifying any additional options, by calling them as simple binary using its program name.

#### Application Monitor agent

```
appMonAgent [-c <inputfile>]
             [-t <int>]
             [-r <old-file> <new-file>]
```

##### **-c** <BS2:inputfile>

You may specify a configuration file when you start the Application Monitor agent (see [section "Statements for the configuration file"](#)). If no configuration file is specified, all the subsystems recognized by the BS2000 system when the Application Monitor agent was started are monitored. The configuration file, which is defined by *<inputfile>*, must be stored in the BS2000 file system.

##### **-t** <int>

Interval at which the agent checks for requests from the command program. The default interval is set to five seconds.

File monitoring is carried out when the interval has expired.

The monitoring interval for subsystems is calculated as five times the value set for the timer interval, i.e. 25 seconds in the standard case.

It may happen that changes of state in applications or job variables may not be notified until the timer interval has expired.

The monitoring interval for DCAM applications is calculated as 60 times the value set for the time interval, i.e. 5 minutes in the standard case.

##### **-r** <BS2:old-file> <BS2:new-file>

Converts the "old" configuration file (existing in SDF format) to the configuration file *<new-file>* with the new format.

## Console Monitor agent

```
consoleAgent -o <operid>
              -y <op-role1> [,<op-role2>, ....., <op-role10>] | -a <auth-file>
              [-k <password>]
              [-c <BS2:msg-filter>]
              [-n <BS2:negative-msg-filter>]
              [-r <proc>]
```



The agent needs credentials for console logging. These can be passed in two ways:

- directly via the options *-o*, *-y* and *-k* (optional)
  - or via the authentication file using option *-a*
- i.e. you have to specify either *-a* or *-o*, *-y* and optionally *-k*.

The Console Monitor agent has further functions, see [Additional functions of the Console Monitor agent](#).

*Description of the options:*

**-o** <operid>

defines the operator ID

Mandatory operand if *-a* is not specified.

**-y** <op-role1> [,<op-role2>, ....., <op-role10>]

Name of the operator role containing the relevant routing code for console monitoring.

Mandatory operand if *-a* is not specified.

**-a** <auth-file>

Using *-a* <auth-file> you can alternatively specify the complete path name of the authentication file where the role(s) and the password are defined.

Example: */etc/snmp/.conMonAuth*.

Mandatory operand if *-y* and *-o* are not specified.



<auth-file> must be created before the agent is started. This can be done by the *-A* option, see below.

**-k** <password>

Definition of the password which authorizes the agent to access \$CONSOLE. No specification (default value) means that no password is required.

*-k* may not be specified together with *-a*.

**-c** <BS2:msg-filter>

Path name of the file containing the filter configuration of the agent. No specification (default value) means that all messages will generate trap.

**-n** <BS2:negative-msg-filter>

Path name of the file containing the negative filter configuration of the agent. No specification (default value) means that no message will be filtered out.

#### *Additional functions of the Console Monitor agent*

– consoleAgent -A <auth-file>

Outputs a prompt with which an authentication file can be generated. This file remains encrypted with AES.

– consoleAgent -s <auth-file>

Outputs the operator IDs and role(s) defined in the authentication file. If a password has been defined, \*SET is output, otherwise \*NONE.

<auth-file> is the absolute path to the authentication file.

### **HSMS agent**

Starting the HSMS agent in the POSIX shell:

```
hsmsAgent -l <HSMS-library>
```

*Description of the option:*

**-l** <HSMS-library>

Path name of HSMS-SYSLIB, mandatory option.

### **openUTM agent for monitoring UTM applications**

```
utmAgent [-c <BS2:inputfile>]
```

*Description of the option:*

**-c** <BS2:inputfile>

When the agent is started, a configuration file can be specified (see [section “Configuring the openUTM agent”](#)).

**Storage agent**

Starting the storage agent in the POSIX shell:

```
storageAgent [-c <BS2:inputfile>]
```

*Description of option:*

**-c** <BS2:inputfile>

BS2000 configuration file describing the pubsets, disks or ROBAR applications are to be monitored.

## 4.4 Starting BCAM, FTP and SESAM/SQL agents manually

These agents are part of the respective products and are supplementing functions of SNMP-AGENTS product as follows:

- `bcamAgent` - agent for monitoring and managing openNet Server parameters
- `ftpAgent` - agent for monitoring and managing FTP servers
- `sesAgent` - agent for monitoring SESAM/SQL databases

All of the presented agents don't have any agent-specific options, therefore they could be ran as the simple binary.



It is not recommended to start agents manually. The best practice is to use rc scripts to perform start/stop of the SNMP products.



---

## 5 NET-SNMP functions

### 5.1 Support of MIB-II (RFC 1213)

NET-SNMP supports the SNMP management defined in RFC 1213 using the following MIB-II groups:

- System group for monitoring the system
- SNMP group for SNMP monitoring
- Interface group
- IP group
- ICMP group
- TCP group
- UDP group

It also provides a series of standardized and proprietary MIBs for SNMP administration.

in addition, NET-SNMP supports individual objects of other MIBS which are relevant for SNMP management in BS2000.

The values (definition, access, ...) for the MIB groups can be displayed via a MIB browser.

## 5.2 Other MIBs supported by NET-SNMP

### SNMP framework MIB (SNMP engine)

The SNMP engine is defined in RFC 2271.

### Objects of other MIBs supported by NET-SNMP

Besides the MIBs for the system and SNMP management, NET-SNMP also supports the other objects of the MIBs relevant for SNMP management in BS2000 systems listed below.

#### *Standardized MIBs*

<b>MIB</b>	<b>root OID of the MIB</b>
SNMP-MPD-MIB	snmpMPDMIB
SNMP-TARGET-MIB	snmpTargetMIB
SNMP-NOTIFICATION-MIB	snmpNotificationMIB
NET-SNMP-EXTEND-MIB	netSnmExtendMIB
SNMPv2-MIB	snmpMIB
IF-MIB	ifMIB
NOTIFICATION-LOG-MIB	notificationLogMIB
DISMAN-EVENT-MIB	dismanEventMIB
DISMAN-SCHEDULE-MIB	schedMIB
NET-SNMP-AGENT-MIB	netSnmAgentMIB
SNMP-USER-BASED-SM-MIB	snmpUsmMIB
SNMP-FRAMEWORK-MIB	snmpFrameworkMIB
SNMP-VIEW-BASED-ACM-MIB	snmpVacmMIB
NET-SNMP-VACM-MIB	netSnmVacmMIB

## 5.3 Functionality of the Event services

The Event services implements the Event MIB (RFC 2981). The Event MIB is divided into the following two sections, which define the Event services range of functions:

- Trigger section
- Event section

These sections are configured with tables in the *snmpd.conf* file according to the directives specified in [section “DisMan Event MIB”](#).

### Trigger section

The trigger section defines the MIB objects to be monitored and the conditions (trigger tests) with which an event is triggered, e.g.

- **mteTriggerValueID** specifies the OID of the MIB object to be checked.
- **mteTriggerTest** specifies the conditions to be tested, e.g.
  - existence: whether the MIB object specified in TriggerValueID exists.
  - boolean: whether the value of this MIB object matches a value defined in the BooleanTable (TriggerBooleanValue).
  - threshold: whether this value is above or below a threshold value defined in the ThresholdTable.
- **mteTriggerSampleType** specifies whether the reference value is to be interpreted as an absolute value (absolute) or as a difference (delta) from a value determined during a previous query.
- **mteTriggerFrequency** specifies the time interval (in seconds) between two contiguous queries.

Depending on the type of test, entries may require a further table:

- **mteTriggerExistenceTable**: Object exists / disappears / changes value.
- **mteTriggerBooleanTable**: Benchmark test of the object or delta value with the reference value.
- **mteTriggerThresholdTable**: Object or delta value falls above / below threshold value.

If the test result is positive, a search is made in the EventTable for an appropriate entry for the action to be carried out.

## Event section

The Event section defines which action - SNMP trap and/or SNMP SetRequest - is to be triggered in response to a successful trigger test, in the object EventAction:

- **notification** (SNMP trap): The OID of the trap to be sent is defined in the EventNotificationTable.
- **set** (SNMP SetRequest): The OID of the MIB object and the value to be set are defined in the EventSetTable.

## Notifications

The Event MIB offers the following traps, which can be sent in response to triggered events.

- **mteTriggerFired** reports that the trigger monitoring an object has been triggered.
- **mteTriggerRising** reports that the threshold value has been exceeded.
- **mteTriggerFalling** reports that the value has fallen below the threshold value.

Objects passed with a notification are entered in the **Objects Table** and specified in the corresponding entry in the:

- Trigger Table
- ExistenceTable / Boolean Table / Threshold Table and / or
- Notification Table

The following notifications should be used only to help diagnose a problem, that has appeared in the error counters and cannot be found otherwise:

- **mteTriggerFailure** reports that an attempt to check a trigger has failed.
- **mteEventSetFailure** reports that an attempt to do a set in response to an event has failed.

## 5.4 Functionality of the Scheduling services

Scheduling services implement the Scheduler MIB (RFC 2591), which supports the following types of scheduling:

- Periodic scheduling
- Scheduling on the basis of calendar dates
- One-shot scheduling

Scheduling services configuration should be performed in the *snmpd.conf* according to the directives from [section “DisMan Schedule MIB”](#).

### Periodic scheduling

Periodic scheduling is based on specific time intervals between two consecutive SNMP set operations initiated by the Scheduler services. A time interval is defined by the number of seconds elapsing between two continuous SNMP statements.

### Scheduling on the basis of calendar dates (calendar scheduling)

Scheduling on the basis of calendar dates initiates actions on specific week days or days in a month. A calendar time is specified by giving the month, day, weekday, hour and minute.

You can specify a number of values for each date and thus define complex scheduling operations. The scheduling can for instance initiate a specific action every 15 minutes on a given weekday.

Date specifications based on months, days and weekdays can be defined using the following BITS type scheduler MIB objects:

- *schedMonth*
- *schedDay*
- *schedWeekDay*

Setting several bits in one of these MIB objects has the effect of a logical OR. If, for instance, you set the bits Monday (1) and Friday (5) in *schedWeekDay*, the scheduling initiates the actions specifically on Mondays and Fridays.

The cross-object combination of bit fields from *schedMonth*, *schedDay* and *schedWeekDay* has the effect of a logical AND. If, for instance, you set the bits June (5) and July (6) in *schedMonth* and the bit fields Monday (1) and Friday (5) in *schedWeekDay*, the scheduling is limited to initiating actions exclusively on Mondays and Fridays in the months of June and July.

When specifying dates, you can implement wildcard functionality if you set all bits to "1".

### One-shot scheduling

One-shot scheduling is similar to scheduling on the basis of calendar dates. The only difference is that one-shot scheduling is automatically deactivated once an action has been initiated.

### Actions

Actions initiated by scheduling model SNMP set operations on MIB objects whose OID is configured in the object *schedVariable*. The value to be set is specified in the object *schedValue*. Actions defined in this MIB are limited to INTEGER type objects. This restriction does not however reduce the usability of the scheduler MIB. Simple scheduling is thus possible, e.g. scheduling the activation/deactivation of resources with a corresponding status MIB object (e.g. *ifAdminStatus*).

---

## 6 SNMP-AGENTS functions

The agents from SNMP-AGENTS implement the functionality described by the following MIB files:

- FJ-Application-Monitoring-MIB.txt
- FJ-Console-Monitoring-MIB.txt
- FJ-HOST-RESOURCES-MIB.txt
- FJ-HSMS.txt
- FJ-OPENFT-MIB.txt
- FJ-OPENSMS2-MIB.txt
- FJ-SPOOL-MIB.txt
- FJ-Storage-Management-MIB.txt
- FJ-UTM-MIB.txt

The MIBS for the agents are located in the directory `/usr/share/snmp/mibs` and can be displayed via a MIB browser and thus are not described in detail.

## 6.1 Application Monitor agent

The Application Monitor agent allows monitoring of:

- user applications
- BCAM applications
- DCAM applications
- subsystems
- job variables and
- log files

Logically associated components of a process (applications, log files, subsystems and job variables) can be monitored together as a group.

The term applications are taken to mean programs and tasks here. The type and extent of application monitoring is controlled individually with the configuration file. Please see the corresponding section on [section “Statements for the configuration file”](#) for information on configuration file creation.

### Log files

Monitoring via log files is provided for those applications that cannot send a trap to the management station themselves. Instead of this, these applications deposit messages in a log file that is monitored by the Application Monitor. The Application Monitor agent evaluates these messages and sends filtered messages to the management station as a trap.

BS2000 log files must be of type ISAM and must be shareable (SHAREUPD=YES). NFS or POSIX log files can be in ASCII or EBDIC format. The default format is EBDIC, ASCII format must be marked in the configuration file. The file name entry in the configuration file must contain the user ID for BS2000 or the absolute path name in the case of NFS / POSIX. The agent cannot otherwise discriminate between BS2000 and NFS / POSIX files.

By default, log files are checked every 5 seconds by the agent. This value can be changed in the start command using the `-t` option. If the agents detect changes to the files, a trap is sent to the management station for new messages, if they are matching the given pattern. When no pattern was specified, all message will trigger traps.



## 6.2 Console Monitor agent

The Console Monitor agent is the agent for monitoring the console interface. It is used for acquiring console messages and entering console commands.

### 6.2.1 Acquiring console messages

Console messages are received by the Console Monitor agent and sent to the management station singly with the computer name and time as trap. The number of messages you will have to deal with will depend on the number, utilization and size of the computers monitored by the Console Monitor agent. It will, however, very seldom be meaningful to pass all console messages on to the management station. The Console Monitor agent therefore provides two options for filtering console messages. Positive and negative message filters are provided.

#### Positive message filter

1. Each console message is assigned a specific routing code. You define the messages to be output on the management station by selecting specific routing codes which are defined in operator roles.
2. The message codes of console messages, query or TYPE I/Os can also be used as selection criteria to define which messages are sent to the management station. The relevant message codes are stored in a message filter file and this is evaluated by the Console Monitor agent when it is started and during a session if the file is updated

#### Negative message filter

The Console Monitor agent allows you to suppress certain messages when logging on to UCON.

Creating the message filter file, which must be specified when the agent is started, is described in [section “Agents-specific options for starting agents manually”](#). The message filter file can be modified during a session by writing the *consMonMsgFilter* MIB object (positive message filter) and via the command program; the negative message filter *consMonNegMsgFilter* cannot be modified during operation.

If the newly assigned message filter file cannot be opened, it is rejected with the return code *General error* and the old file is used. No traps are sent to the management station if the message filter file contains either no message codes or no valid ones. The negative message filter *consMonNegMsgFilter* cannot be modified during a session.

## 6.3 Host Resources agent

The Host Resources agent supplies information about the system, devices, and file system, as well as about the installed software according to Standard RFC 1514.

## 6.4 HSMS agent

The HSMS agent allows you to read and change global HSMS data. It also supplies detailed information on HSMS requests and their states. You can restrict the scope of the information displayed using the selection criteria "state" and "origin". The HSMS agent itself does not send any traps.

All HSMS requests that are processed by the respective BS2000 host are displayed in a table. The HSMS agent determines this information by evaluating an OPS variable.



To ensure that the agent can continue displaying the requests even after they have terminated, the requests must not be deleted by command. Requests with the status *COMPLETED* are, however, deleted automatically by an implicit recovery at the start of each HSMS session.

The number of requests displayed can be restricted depending on the

- processing state of the requests
- host from which the request originates

## 6.5 openFT agent

The file transfer agent is used for

- starting and stopping openFT (BS2000)
- acquiring system parameter information
- changing the encryption public key
- statistic data output
- diagnosis control
- outputting partner information

FT is started and stopped via the openFT agent by setting the value of *ftStartandStop.0* to *START* and *STOP* respectively.

## 6.6 openSM2 agent

The performance agent for openSM2 supplies information on openSM2 itself, i.e. on subsystem status, version, measurement interval size and sampling cycle.

The actual measurement values correspond to the SM2 report groups and provide information on

- CPU utilization
- I/O activities
- main memory and virtual address space utilization
- main memory occupation by the four standard task categories
- input/output operations to peripheral devices during a measurement interval
- application-specific data of UTM applications
- resource usage of separate tasks
- measurement values for VM2000 systems

## 6.7 openUTM agent

The openUTM agent provides the following services:

- monitoring and controlling selected openUTM applications
- information on system parameters, physical and logical terminals, terminal pools, transaction codes, transaction classes, user data, connections and statistic data
- modifying application properties and system parameters
- locking/unlocking UTM data terminals
- changing the configuration file
- terminating an UTM application

## 6.8 Spool agent

The SPOOL and RSO devices are monitored by the spool agent, which supplies information about devices and print jobs. The spool agent is supplied with a proprietary MIB that consists of the device group and the job.

## 6.9 Storage agent

The agent for storage management supplies information about pubsets and disks as well as on the availability of the storage management products HSMS, MAREN and ROBAR. Correspondingly, a proprietary MIB is supplied with the agent, which contains the following information in addition to the global data of the storage management agent:

- general information about HSMS, MAREN and ROBAR
- resource information,
- display of all pubsets in a table
- display of all disks in a table

There are configurable monitoring possibilities for the following parameters (see [section “Configuring the Storage agent”](#)):

- changes of pubset saturation level
- state of disk reconfiguration

---

## 7 BCAM, SESAM/SQL and FTP agents functions

This chapter describes the functionality of the BCAM, SESAM/SQL and FTP agents. FTP is part of interNet Services and delivered via TCP-IP-AP.

### 7.1 BCAM agent

The BCAM agent implements a private MIB FJ-BCAM-MIB which defines monitoring of:

- BCAM memory usage
- BCAM trace setting
- Applications configured in BCAM
- Connections
- Routers, routes and network connections
- Hosts
- Mapping of addresses to applications

## 7.2 FTP agent (from TCP-IP-AP package)

The FTP agent implements a private MIB FJ-FTP-MIB, which allows monitoring and controlling the FTP server. The following information is provided:

- notification about the start or shutdown of an FTP server
- server-specific data about various parameters and statuses of the server
- connection data

The following server controlling options are available:

- start the FTP server
- set the FTAC level (when the FTP server is started)
- shut down the FTP server
- activate/deactivate a socket trace
- activate/deactivate debugging
- save the log file
- increase the maximum possible number of parallel connections
- modify the timeout value for connections
- set the FTAC job class (FTACJob)

## 7.3 SESAM/SQL agent

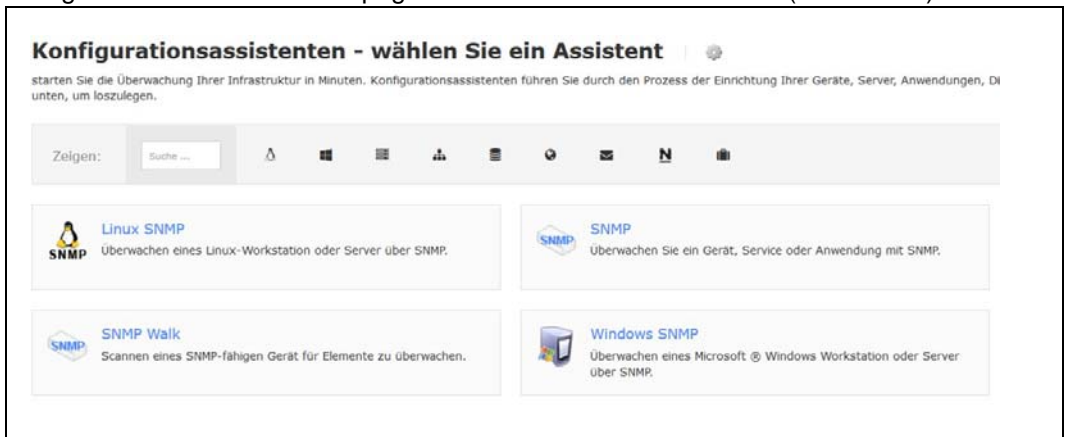
The SESAM/SQL agent supplies information about SESAM/SQL databases, DBHs and DCNs, which are used to process these databases.

## 8 Example for operating the management station

This chapter provides an integration Example in NagiosXi. It describes how a value from a BS2000 server can be read out and monitored via SNMP.

A simple example will be shown, because there is large number of values and different parameters for each of these values.

Go to the NagiosXI URL and use the *Konfigurieren (Configure)* tab to navigate to the configuration wizard overview page. There choose the *SNMPWizard (SNMPWizard)*:



The screenshot shows the 'Konfigurationsassistenten - wählen Sie ein Assistent' page. It features a search bar and a navigation menu. Below the menu, there are four cards representing different SNMP monitoring options: Linux SNMP, SNMP, SNMP Walk, and Windows SNMP. Each card includes an icon and a brief description of the monitoring capability.

Define the BS2000 to be monitored using the IP address (*Geräte-Adresse*).



The screenshot shows the 'Configuration Wizards: SNMP - Schritt 1' page. It displays the 'SNMP-Informationen' section with a text input field for 'Geräte-Adresse' containing the value 'xxx.xxx.185.51'. Below the input field, there is a note: 'Die IP-Adresse oder den vollqualifizierten DNS-Namen des Servers oder Gerät, das Sie gerne zu überwachen hatte.' At the bottom, there are two buttons: 'Zurück' and 'Nächste'.

## Example for operating the management station

If required/desired choose an alias/display name for the server:

 **Configuration Wizards: SNMP - Schritt 2** 

---

Device Details

Geräte-Adresse:

Host Name:


Der Name, den Sie gerne mit diesem Server oder Gerät zugeordnet haben würde.

Select the SNMP version, the port and the SNMP community:

**SNMP-Einstellungen**

---

Geben Sie die Einstellungen auf den Server oder das Gerät per SNMP überwachen.

SNMP Version:  

Das SNMP-Protokoll-Version verwendet werden, um mit dem Gerät communicate.

HTTP Port::

Der zu verwendende SNMP-Port ist Port 161.

---

**SNMP-Version Einstellungen**

---

SNMP-Community:


Der SNMP-Community-String verwendet, um das Gerät abzufragen.

From the Management Information Base (MIB) select the required/desired object (s) using the Object Identifier (OID):

**SNMP-Dienste**

---

Geben Sie alle OIDs Sie gerne über SNMP zu überwachen hatte. Beispielinträge wurden als Beispiele zur Verfügung gestellt.

OID	Display Name	Datenbeschriftung	Data Units (Option)	Spiel Typ	Warnung Reichweite	Critical Reichweite
<input checked="" type="checkbox"/> <input type="text" value=".1.3.6.1.4.1.231.2.20.3.10.:"/>	<input type="text" value="PubsetSaturationLevel"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="Numerisch"/> 	<input type="text" value="3"/>	<input type="text" value="4"/>

[Add Row](#) | [Delete Row](#)



Choose the monitoring interval:

**SNMP Configuration Wizards: SNMP - Schritt 3**

**Überwachung-Einstellungen**

Definieren grundlegender Parameter, wie die Host- und Service (s) überwacht werden sollten bestimmen.

**Unter normalen Umständen:**

Überwachen Sie die Host- und Service (s) jeden  Minuten.

**Wenn ein Potential Problematik ist zunächst erkannt:**

Überprüfen Sie erneut die Host- und Service (s) jeden  Minuten bis zu  Zeiten vorher [Senden einer Benachrichtigung](#).

Set, if someone should be notified in case of problems:

**Notification Settings**

Definieren grundlegender Parameter, wie Benachrichtigungen für den Host- und Service (s) geschickt werden sollen bestimmen.

**Wenn ein Problem erkannt wird:**

Senden Sie keine Benachrichtigungen

Senden Sie eine Benachrichtigung sofort

Warten Sie  Minuten, bevor eine Benachrichtigung

**Sollten die Probleme fortbestehen:**

Senden Sie eine Benachrichtigung jeden  Minuten, bis das Problem behoben ist.

Specify the people and groups, who need to be notified:

**Senden Sie Warnmeldungen an:**

Mich (Passen Sie die Einstellungen)

Andere individuelle Kontakte

Default Contact [x1\_default\_contact]

Spezifische Kontaktgruppen

All Contacts [x1\_contactgroup\_all]

hages Administrators (admins)

Optional service groups can be used and host parents can be selected:

Service Groups

Definieren, welche servicegroup (s) die überwachte service (s) gehören soll (falls vorhanden):

Host-Eltern

Definieren Sie, welche Host (s) werden als die Eltern des zu überwachenden Host (Host vorhanden). Anmerkung: In der Regel nur ein (1) als Host übergeordnet angegeben.

- all (keine) host.name (172.25.81.7)
- all (keine) host.name (172.25.81.13)
- all (keine) host.name (172.25.81.9)
- localhost (127.0.0.1)
- SU Augsburg (172.17.185.51)

Click on the bottom *Anwenden (Apply)*, to save configuration:

SNMP Configuration Wizards: SNMP - Endschritt

Endgültigen Einstellungen

Klicken **Anwenden** Ihre neue Konfiguration hinzuzufügen.

[Zurück](#) [Anwenden](#) [Service Template](#)

Under *KonfigurierenCore Config Manager Dienstleistungen (Configure Core Config Manager Services)* you can check the configuration. Click on the bottom *Test Ankunft Befehl (Test arrival command)*.

Config Name \*

SU Augsburg

Beschreibung \*

PubsetSaturationLevel

Anzeigenamen

Hosts verwalten

Vorlagen verwalten

Verwalten Hostgruppen

Verwalten Servicegruppen

Aktiv

Prüfbefehl

check\_xi\_service\_snmp

Command View

\$USER1\$/check\_snmp -H \$HOSTADDRESS\$ \$ARG1\$

\$ARG1\$ -p 1161 -o .1.3.6.1.4.1.231.2.20.3.10.1.14.4.86.49

\$ARG2\$

\$ARG3\$

\$ARG4\$

\$ARG5\$

\$ARG6\$

\$ARG7\$

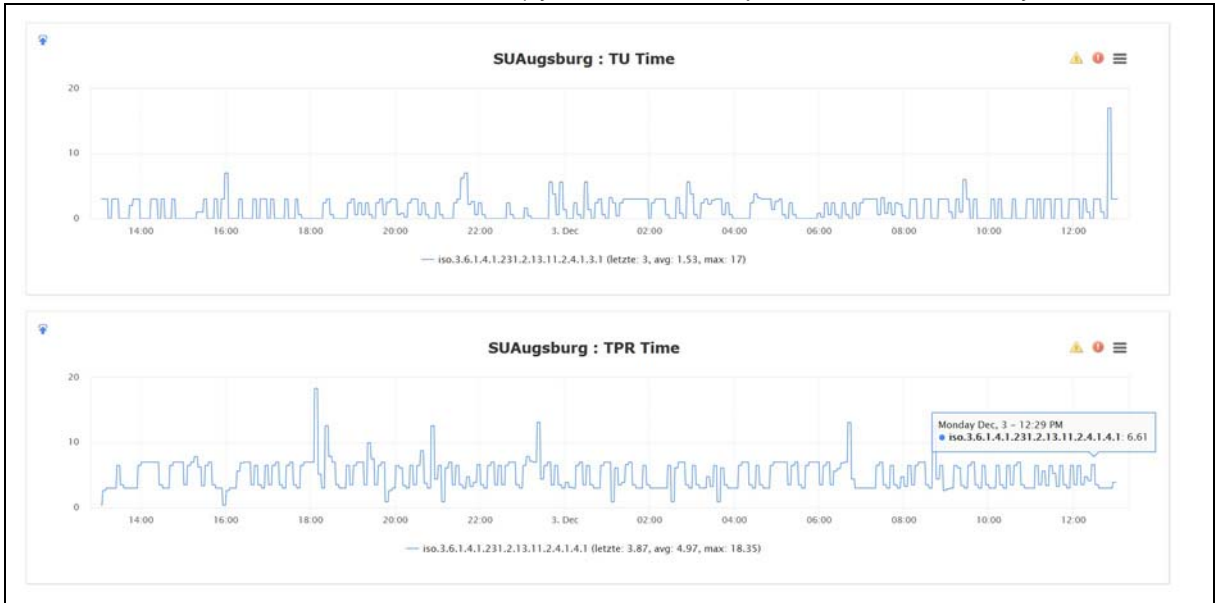
\$ARG8\$

Test Ankunft Befehl

If the command is processed correctly, SNMP OK and an appropriate value should be returned.

What the individual values mean, can be looked up in the MIB.

Under *ZuhauseHost Status Hosteintrag - Leistungsdiagramme (Graphsymbol)* (*HomeHost Status Host Entry - Performance Diagrams (graph icon)*) you can view the performance charts for your host:



Using the blue arrows in the upper left corner, these can be attached to dashboards.



---

## 9 Appendix: Procedure in the event of errors

Each agent keeps trace files in which error messages are logged as standard.

### 9.1 Format of the logging entries

By default, each agent stores its logging entries in the file */var/adm/syslog*. The entries of this logging file have the following basic format:

```
<date> <time> keyword <daemon>[pid]: <message text>
```

**<date> <time>**

Time stamp of the logging entry.

**keyword**

Keyword for classifying the message (e.g. LOG\_INFO, LOG\_NOTICE, LOG\_DEBUG and LOG\_ERR).

**daemon[pid]**

Process name (daemon name) of the agent with the information on the corresponding process ID (PID). If there is no valid PID available for the message output (e.g. in the case of a message output during starting the daemon), an empty bracket expression is output.

**message text**

Plain text of the message.

## 9.2 Configuring logging files of agents

Logging of agents is configurable and will be performed to the POSIX file which is specified when starting the agent.

*Examples:*

- `bcamAgent -Lf /var/adm/bcam.log`  
will start `bcamAgent` and write the logging entries to the `/var/adm/bcam.log` file.
- `bcamAgent -Ls d`  
will start `bcamAgent` and write the logging entries to the default logging file `/var/adm/syslog` with daemon facility.

## 9.3 Debug options

All SNMP related products are supporting the debug arguments, which could be configured either by `snmp.conf` (see [section “Client behavior”](#)) or passed by the command line arguments `-D[TOKEN[,...]]`.

The following tokens available:

- in all SNMP products:

- agentx
- init\_mib
- mib\_init
- parse-file
- parse-mibs
- read\_config
- recv
- snmp
- transport
- trap

- in snmpd only:

- disman
- dumpv
- dumph
- exec
- mteTrigger
- run
- snmpd

snmp1s  
 snmpusm  
 snmpv3  
 usm  
 usmUser

- in SNMP-AGENTS as well as in BCAM, FTP and SESAM/SQL agents:  
 DEBUG (be careful, as it's printing a lot of diagnostic information)

Also, each agent has its separate debugging token, which can be found in the next table:

Agent	Token
appMonAgent	appMon
bcamAgent	bcam
consMonAgent	console
spoolAgent	spool
	fjSpool
ftpAgt	ftp
	ftpAgent
hostAgent	host
hsmsAgent	hsms
openFTAgent	openFT
openSM2Agent	openSM2
sesAgent	sesam
storageAgent	storage
utmAgent	utm





---

# Glossary

## agent

The agent is also known as the management agent. It is the implementation of a management protocol that exchanges management information with a management station. An agent is a program that runs on a system or device and reports the current information about the system/device to a manager or corresponding management application.

## alarm

A group of states and state transitions. The states correspond to entities of an object type with attribute values that are specified by the network administrator. Whenever the monitored object type of a device or line changes into a state that the administrator has defined as an alarm state, the management platform reports the event by displaying a corresponding icon and changing the color of the alarm and device icons.

## attribute

An attribute is part of an object type definition in an MIB module. It defines one characteristic in an object type. If an object type contains more than one instance, the attributes define the columns and the entities the rows in the object type table. The table entries are the instance values for the attribute. See also *object type* and *object instance*.

## characteristic

This can be either an object type or a characteristic string. Both can belong to one characteristic group. A characteristic string is a characteristic (but not an object type) that is added to a characteristic group by the manufacturer or network administrator for the purpose of restricting the scope of polls and alarms. A characteristic string defines the menu and submenus that are available in the device overview under the *objects* button. It also defines the applications that are available in the device view under the *applications* button.

## community string

A simple password that is shown in the network map when a device icon is added. The agent that runs on the device requires this password from the manager before information about the device is made available.

**connection**

The object instance that describes a (line) connection to a network management device.

**connection instance**

An object instance of a connection to a device. See *object instance*.

Both ends of a line icon can be assigned to one device. This connection has two aspects. Firstly, it is a graphical representation of part of the physical network; secondly, it is an object type of the device (e.g. an object type for connection or junction information).

**device**

A network system, router, hub or other addressable equipment within the network, but not a line, tap or network map icon.

**event report**

Event reports indicate errors, state changes and similar important events in the system. They occur asynchronously (spontaneously), are command-independent, object-oriented and are delivered to an arbitrary network management station within the network, according to the requester principle.

**gateway**

A gateway connects heterogeneous networks.

**HTML**

HTML (HyperText Markup Language) is a standardized markup language that consists of a subset of the SGML Standard (Standard Generalized Markup Language). HTML documents can be exchanged between any computer systems using the standardized HTTP communications protocol.

**HTTP**

HTTP (HyperText Transfer Protocol) is the communications protocol used between systems in the World Wide Web (WWW). HTTP enables documents to be exchanged between any computer systems / applications.

**Internet**

Communication architecture characterized by the user of TCP and IP, having developed from the ARPA network in the USA. Extensions are controlled by the IAB with the aid of the RFC process.

**IP address**

Representation of a connection point in the Internet:  
IPv4: 4 bytes (32 bits), IPv6: 16 bytes.

**major trap number**

The SNMP Standard (RFC 1157) defines seven trap categories with the numbers 0 to 6. These numbers are designated as major trap numbers.

**management station**

A system in the network on which a management application runs.

**manufacturer-specific extensions**

Additional SNMP management objects for a device that are supplied by a manufacturer for the agents of this device. They are also frequently known as manufacturer's MIB.

**MIB**

MIB stands for "Management Information Base". This designates a data model that describes the network elements to be administered with network management (managed nodes), in an abstract form. This data model consists of the formal descriptions of object types (object classes) that are constructed according to the RFC 1157 conventions.

**MIB-II**

MIB-II is a standard MIB whose use is obligatory in the Internet. It offers an adequate data model for managing devices. MIB-II is standardized and is defined in RFC 1213. It is an extension of MIB-I (RFC 1156).

**network management protocol**

The protocol for exchanging management information.

**network map**

A collection of lines and icons that are arranged into a group of interconnected network maps. The corresponding network map background maps that display a network and its subnetworks are optional.

**network map file**

A text file that contains the configuration information for its network: the file names of background maps for network and subnetwork maps; the file names and positions of icons for systems, routers, hubs and lines; configuration information for polls, masks and alarms; characteristic groups. This file is also called the " Map Database" or "Map\_db" file.

**network map icon**

An icon that displays a network map in a group of nested network maps. The icon is displayed in the next higher network map. Network map icons can also be user-defined.

**object**

In an MIB: an object type or attribute.

On the graphical interface: device, line, tap, poll, mask or alarm, or a specific instance of this.

**object instance**

Represents the characteristics (attribute values) of a device. The entities are managed by the device agents.

The object instance is specified by the instance identifier or index.

**object identifier (OID)**

A notation that designates the position of an object in an MIB tree. For example, 1.3.6.1.4.1.231.1.3.2 (*iso.org.dod.internet.private.enterprise.fj.1.3.2*) specifies an RM600 system. There are also MIB names for the object identifier (e.g. *cisco* for a Cisco router).

**object type**

A class of object entities of the same type, that is defined by a formal description. There may be one or more entities on one device for an object type. The object type is in the form of a table if more than one instance is possible for an object type on one device. Each row of this table represents one object instance and the columns the attributes of the object type.

Object class is another name for object type.

**ping**

A protocol with which the IP levels connectivity from one IP address to another is checked.

**poll**

Cyclic request for information about MIB object types. The network administrator can carry out configuration.

**poll cycle**

The poll cycle is the parameter that defines how often SNMP contacts an agent on a device in order to call up information from the MIB of this device.

**protocol**

A number of rules with which systems communicate with each other

See also *SNMP* and *ping*.

**RFC**

Request for comments. The series of documents that describe the Internet protocol and related standards.

**SNMP**

SNMP stands for "Simple Network Management Protocol". SNMP is a standard protocol for network management in TCP/IP networks.

**state**

Alarm state: an element in an alarm definition (see *alarm*).

MDC state: the *Domain Table View* window shows a code under the *state* entry. This code describes whether a local or remote Client Manager sends or receives a domain.

**state transition**

Alarm change of state that is activated by a trigger.

**subnetwork**

A physical network within an IP network.

**subnetwork icon**

An icon in a root or subnetwork map that represents a nested subnetwork map one level below the current network or subnetwork map.

**tap**

A tap represents the connection point between a device and the network in a network map. A tap can be created, configured and deleted, but it cannot be managed.

**TCP/IP**

TCP/IP stands for "Transmission Control Protocol/Internet Protocol", i.e. the Internet protocol. A number of rules that define how systems communicate with each other in an open (not manufacturer-bound) environment. This is normally a large communication infrastructure (Internet).

**trap**

Under SNMP, traps are problem reports that are sent automatically by an agent.

**trigger**

A trigger is a message that is sent by the poll or mask system to the alarm system. An alarm executes a state transition when a specific trigger is received.

### **URL**

URL (Uniform Resource Locator) is a character string which users enter in their Web browser to access a WWW document.

The URL for the WWW contains the address of the required Website and consists of the following components: protocol, computer address (host domain name or IP address), port number if required, path and file name if required, and (optionally) details of a place within the document text.

### **variable**

Under SNMP, a variable is the result of linking an object instance name with an assigned value.

---

## Related publications

You will find the manuals on the internet at <http://manuals.ts.fujitsu.com>. You can order printed copies of those manuals which are displayed with an order number.

### BS2000 manuals

**openNet Server V4.0**  
**BCAM V24.0A Volume 1/2**  
User Guide

**openNet Server V4.0**  
**SNMP-Management for openNet Server**  
User Guide

**interNet Services V3.4B (BS2000)**  
Administrator Guide

**interNet Services V3.4B (BS2000)**  
User Guide

**BS2000 OSD/BC V11.0**  
DSSM V4.3  
Subsystem Management  
User Guide

**HSMS (BS2000)**  
HSMS Functions  
User Guide

**openFT (BS2000)**  
**Command Interface**  
User Guide

**openFT (BS2000)**  
**Installation and Operation**  
System Administrator Guide

**openUTM**  
**Generating Applications**  
User Guide

**openSM2**  
**Software Monitor**  
User Guide

**SPOOL V4.6A (BS2000)**  
User Guide

**RSO V3.5A / V3.6A(BS2000)**  
**Remote SPOOL Output**  
User Guide

**SESAM/SQL-Server V9.1 (BS2000)**  
Database Operation  
User Guide

**POSIX (BS2000)**  
Basics for Users and System Administrators  
User Guide

**IMON V3.2 (BS2000)**  
**Installation Monitor**  
User Guide

## Other related publications

Douglas Steedman  
**Abstract Syntax Notation One (ASN.1): The Tutorial and Reference**  
Isleworth, 1990  
(ISBN 1-871802-06-7)

Marshall T. Rose  
**The Simple Book: An Introduction to Management of TCP/IP-based Internets**  
Prentice-Hall  
(ISBN 0-13-812611-9)



## Ordering RFCs

If the Requests for Comments (RFCs) referred to in the text are available under the URL <https://www.rfc-editor.org/>.



---

# Index

## A

access control, MIB objects 26  
actions (Scheduler services) 118  
ADD-APPLICATION-RECORD  
    statement for the Application Monitor 72  
    statement for the openUTM agent 95  
ADD-DCAM-APPLICATION-RECORD  
    statement for the Application Monitor 73  
ADD-DISK-RECORD  
    statement for the storage agent 91  
ADD-JV-RECORD  
    statement for the Application Monitor 77  
ADD-LOG-FILE-RECORD  
    statement for the Application Monitor 75  
ADD-PUBSET-RECORD  
    statement for the storage agent 90  
ADD-ROBAR-RECORD  
    statement for the storage agent 92  
ADD-SUBSYSTEM-RECORD  
    statement for the Application Monitor 74  
address  
    recipient's 27  
    sender 27  
agent  
    HSMS 122  
agents  
    functionality 23  
    required privilege 107  
application management 17  
Application Monitor agent 20  
    ADD-APPLICATION-RECORD 72  
    ADD-DCAM-APPLICATION-RECORD 73  
    ADD-JV-RECORD 77  
    ADD-LOG-FILE-RECORD 75  
    ADD-SUBSYSTEM-RECORD 74

    change the configuration file in current  
        session 81  
    create configuration file 71  
    DEFINE-OBJECT 79  
    SET-TIMER-OPTIONS 81  
application monitoring  
    control 71  
appMonConfFile  
    change configuration file 81

## B

BCAM application  
    monitor (ADD-APPLICATION-RECORD) 72  
BS2000 log file 120

## C

calendar dates, scheduling 117  
calendar scheduling 117  
communication  
    between SNMP manager and agent 15  
    UTM agent / UTM application 93  
community string 25, 26  
configuration  
    openUTM agent 93  
    storage agent 89  
configuration file  
    create for Application Monitor 71  
    format 71  
    of the Application Monitor, change 81  
    openUTM agent (format) 94  
configuring  
    openUTM agent 94  
consmonagt 109

- consMonMsgFilter
  - Console Monitor [88](#)
  - positive message filter [88](#)
- consMonNegMsgFilter
  - negative message filter [87](#)
- console interface
  - monitor [121](#)
- console message
  - filter [121](#)
  - message code [121](#)
  - routing code [121](#)
- Console Monitor
  - consMonMsgFilter [88](#)
  - filter options [83](#)
  - message filter [84](#)
  - message filter file [84](#)
  - modify configuration file [88](#)
  - QUESTION [85](#)
  - start [109](#)
  - TYPE I/O messages [86](#)
  - TYPIO [86](#)
- Console Monitor agent [20](#)
  - consMonMsgFilter [88](#)
  - consMonNegMsgFilter [87](#)
  - msgid [84](#)
  - name convention (message filter file) [84](#)
- control
  - application monitoring [71](#)
- create
  - configuration file (Application Monitor) [71](#)
  - operator role [83](#)
- D**
- DEFINE-OBJECT
  - statement for the Application Monitor [79](#)
- definition
  - message filter [83](#)
- deleting
  - SINLIB [31](#)
- delivery scope
  - NET-SNMP [33](#)

- E**
- end-of-line
  - BS2000 upicfile [93](#)
- error
  - procedure in the event of [133](#)
- error handling [133](#)
- event section (Event services) [116](#)
- Event services
  - event section [116](#)
  - functionality [115](#)
  - notifications [116](#)
  - Trigger section [115](#)
- example
  - upicfile [93](#)
- F**
- filter
  - console messages [121](#)
- filter options
  - Console Monitor [83](#)
- format
  - configuration file (openUTM agent) [94](#)
  - of configuration file [71](#)
- functionality
  - agents [23](#)
  - Event services [115](#)
  - Scheduling services [117](#)
  - SNMP daemon [22](#)
- G**
- GetNextRequest-PDU [16](#)
- GetRequest-PDU [16](#)
- GetResponse-PDU [16](#)
- H**
- header (SNMP) [16](#)
- host resources
  - MIB [122](#)
- HSMS
  - MIB [122](#)
  - monitoring [124](#)
- HSMS agent [122](#)
  - start [110](#)

**I**

## information

- on installation 31

## Installation

- SNMP-AGENTS 33

## installation

- important information 31
- in BS2000 OSD/BC 31

**J**

## job variable

- monitor (ADD-JV-RECORD) 77

**L**

## log file 120

- for monitoring 120
- monitor (ADD-LOG-FILE-RECORD) 75

**M**

## management

- application 17
- network 17
- system 17

## management architecture (SNMP) 14

## Management Information Base 14

## MAREN, monitoring 124

## message code

- console message 121
- Console Monitor 84

## message filter

- definition 83
- msgid 84
- negative 83
- positive 83
- QUESTION 85
- TYPIO 86

## message filter file

- Console Monitor 84
- name convention 84

## MIB 14

- host resources 122
- HSMS 122
- print service 124
- storage management 124

## MIB object, access to 26

## modify

- configuration file (Console Monitor) 88

## monitor

- with log file 120

## monitoring

- BCAM application (ADD-APPLICATION-RECORD) 72
- console interface 121
- HSMS 124
- job variable (ADD-JV-RECORD) 77
- log file (ADD-LOG-FILE-RECORD) 75
- MAREN 124
- ROBAR 124
- SPOOL device 124
- subsystem (ADD-SUBSYSTEM-RECORD) 74
- user application (ADD-APPLICATION-RECORD) 72

## msgid

- message filter 84

**N**

## name convention

- message filter file (Console Monitor) 84

## negative message filter 83

## NET-SNMP 20

## network and system security 24

## network management 17

## notifications

- Event services 116

**O**

## one-shot scheduling 118

## openUTM agent

- ADD-APPLICATION-RECORD 95
- communication with UTM application 93
- configuration 93, 94
- runtime environment 96

## operator role

- create 83

## overview

- SNMP daemon 20
- SNMP, administrable systems 18

### P

Parallel installation  
    SNMP V6.x and NET-SNMP 23  
PDU 16  
    type 16  
periodic scheduling 117  
positive message filter 83  
privilege  
    for agents required 107  
privileges  
    for starting the agents 107  
procedure in the event of errors 133  
product structure  
    SNMP management for BS2000 OSD/BC 19  
Protocol Data Unit (PDU) 16

### Q

QUESTION  
    message filter 85

### R

rc scripts 102  
Readme file 11  
recipient's address 27  
recommendations  
    network and system security 24  
    using the SNMP service safely 24  
requirements  
    start agents 107  
ROBAR  
    monitoring 124  
routing code  
    console message 121  
    Console Monitor 83  
RSO  
    MIB 124  
RSO device  
    monitor 124  
runtime environment  
    openUTM agent 96

### S

Scheduler services  
    actions 118

    calendar scheduling 117  
    one shot scheduling 118  
    periodic scheduling 117  
scheduling  
    on the basis of calendar dates 117  
    one-shot 118  
    periodic 117  
Scheduling services  
    functionality 117  
security considerations when using SNMP 24  
    recommendations 24  
semicolon  
    BS2000 93  
sender address 27  
SET-TIMER-OPTIONS  
    statement for the Application Monitor  
        agent 81  
SetRequest-PDU 16  
Simple Network Management Protocol 13  
SINLIB  
    deleting 31  
SM2 agent 17  
SNMP 13  
    overview 18  
    security considerations when using 24  
SNMP daemon  
    functionality 22  
    overview 20  
SNMP header 16  
SNMP management  
    architecture 14  
    for BS2000 OSD/BC (product structure) 19  
    of BS2000 OSD/BC 17  
    user interfaces 23  
SNMP protocol elements 16  
SNMP request 25  
SNMP service, using safely 24  
SNMP trap 27  
    community string for 27  
SNMP trap see trap  
SNMP-AGENTS 20  
    software requirements 29  
snmpd 22  
    start 103

- snmpget 105
  - snmpset 105
  - snmptrapd 104
  - snmpwalk 105
  - software requirements
    - SNMP integration 29
  - SPOOL
    - MIB 124
  - SPOOL device
    - monitoring 124
  - start
    - agents (necessary privileges) 107
    - HSMS agent 110
    - rc scripts 102
    - SNMP-daemon 103
    - snmpd 103
  - start Console Monitor 109
  - storage agent
    - ADD-DISK-RECORD 91
    - ADD-PUBSET-RECORD 90
    - ADD-ROBAR-RECORD 92
    - configure 89
    - MIB 124
  - subsystem
    - monitor (ADD-SUBSYSTEM-RECORD) 74
  - SYSROOT 107
  - system management 17
  - system security 24
- T**
- trap see also SNMP trap
  - trap-PDU 16
  - Trigger section (Event services) 115
  - TYPE I/O messages
    - Console Monitor 86
  - TYPIO
    - message filter 86
- U**
- Uninstallation
    - NET-SNMP 34
    - SNMP-AGENTS 34
  - upicfile 93
  - user application
    - monitor (ADD-APPLICATION-RECORD) 72
  - User-based security model 26
  - using SNMP with security in mind 24
  - USM 26
- V**
- varbinds 16, 66
  - variable bindings 16, 66

