

# SNMP Support Add-on for SecurePlatform

## Overview

This add-on adds support for SNMP to SecurePlatform. This support includes:

- Net-SNMP Support for full OS-MIB-II.
- Monitoring of Check Point Status Information (AMON) through SNMP.
- SNMP V.2 and V.3 Support.

## Installation

### Installing Using the local “patch” command

The `patch` command can install packages from TFTP server, or, in the expert mode, from a local file.

To install the SNMP feature on your SecurePlatform machine, proceed as follows:

- If you are installing from TFTP server, run the following command:

---

```
patch add tftp <IP address> SecurePlatformSNMPAddOn.tgz
```

---

- If you are installing from a local file, run the following command:

---

```
patch add <full path to SecurePlatformSNMPAddOn.tgz>
```

---

### Installing Using SmartUpdate

Before you can install the SNMP feature remotely using SmartUpdate, proceed as follows:

- 1** Run `patch` on your SmartCenter Server.
- 2** On a Windows SmartCenter Server, run `splatSNMPAddonMgmtPatch.bat`.  
On a Unix SmartCenter Server, run `splatSNMPAddonMgmtPatch`.

**3** Add the SecurePlatformSNMPAddOn.tgz file to the repository.

After this, you can install the SNMP Add-On on your Secure Platform remotely using SmartUpdate.

## Configuring the SNMP Agent

For basic SNMP configuration use the `snmp` command in the restricted shell, as follows:

---

```
snmp service enable [<portnumber>]
snmp service stat
snmp service disable
snmp user add noauthuser <username> [oidbase <OID>]
snmp user add authuser <username> pass <passphrase> [priv
<privacyphrase>] [oidbase <OID>]
snmp user del <username>
snmp user show [<username>]
```

---

`snmp service enable` starts the SNMP agent daemon listening on the specified UDP port.

`snmp service disable` stops the SNMP agent daemon.

`snmp service stat` displays service status.

`snmp user add` adds an SNMP v3 user to the agent. Authentication and encryption passwords can be specified for the user. Additionally, the user's access can be restricted to the specified OID sub-tree.

`snmp user del` deletes a user. SNMP v1 and v2 users can also be deleted using this command.

`snmp user show` displays a list of existing users.

`snmp user show <username>` displays the specified user's details: access level information and OID subtree restriction.

## Configuring SNMP Traps

SNMP traps can be sent using the `snmptrap` command (in expert mode only).

### **snmpd.conf file**

Additionally, SNMP traps can be configured in the `/etc/snmp/snmpd.conf` file. `snmpd.conf` is the configuration file which defines how the Net-SNMP SNMP agent operates. This file is not required for the agent to operate and respond to requests.

Important `snmpd.conf` directives are described below.

**authtrappable NUMBER**

Setting `authtrappable` to 1 enables the generation of authentication failure traps. The default value is disabled (2). Ordinarily the corresponding object (`snmpEnableAuthenTraps.0`) is read-write, but setting its value makes the object read-only, and attempts to set the value of the object will result in a `notWritable` error response.

**trapcommunity STRING**

This defines the default community string to be used when sending traps. This command must be used prior to any of the following three commands that use this community string.

- `trapsink HOST [COMMUNITY [PORT]]`
- `trap2sink HOST [COMMUNITY [PORT]]`
- `informsink HOST [COMMUNITY [PORT]]`

These commands define the hosts to receive traps (and/or inform notifications). The daemon sends a Cold Start trap when it starts up. If enabled, it also sends traps on authentication failures. Multiple `trapsink`, `trap2sink` and `informsink` lines may be specified to specify multiple destinations. Use `trap2sink` to send SNMPv2 traps and `informsink` to send inform notifications.

If `COMMUNITY` is not specified, the string from a preceding `trapcommunity` directive will be used.

If `PORT` is not specified, the well-known SNMP trap port (162) will be used.

**trapsess [SNMPCMD\_ARGS] HOST**

This is a more generic trap configuration token that allows any type of trap destination to be specified with any version of SNMP. See the `snmpcmd(1)` manual page for further details on the arguments that can be passed as `SNMPCMD_ARGS`.

In addition to the arguments listed there, the special argument `-ci` specifies that inform notifications are to be used instead of unacknowledged traps. This requires that you specify a version number of `v2c` or `v3`.

**agentSecName NAME**

The `DISMAN-EVENT-MIB` support requires a valid user name for which to scan your agent.

This can either be specified using the `agentSecName` token or by explicitly list one on the “monitor” lines described below using the `-u` switch. In either case, a “rouser” line (or equivalent access control settings) must be specified with the same security name name.

**Example**

```
agentSecName internal
```

rouser internal

### monitor [OPTIONS] NAME EXPRESSION

This directive instructs the agent to monitor itself for problems based on EXPRESSION. EXPRESSION is a simple expression based on an oid, a comparison operator (!=, ==, <, <=, >, >=) and an integer value (see the examples below).

NAME is an arbitrary name of your choosing for administrative purposes only.

OPTIONS include the following possibilities:

parameter	meaning
-r FREQUENCY	Monitors the given expression every FREQUENCY seconds. The default is 600 (10 minutes).
-u SECNAME	Use the SECNAME security name for scanning the local host. Specifically, this SECNAME must then be given access control rights via something like the "rouser" <code>snmpd.conf</code> token for this expression to be valid at all. If not specified, it uses the default security name specified by the <code>agentsecname snmpd.conf</code> token. Either the <code>-u</code> flag or a valid <code>agentsecname</code> token must be specified (and that name must be given proper access control rights via a "rouser" token).
-o OID	Specifies additional object values to be delivered with in the resulting trap in addition to the normal trap objects. This is useful for obtaining other columns in the table for the row that triggered the expression. See the examples below for more details.

The following example configuration checks the `hrSWRunPerfTable` table (listing running processes) for any process which is consuming more than 10Mb of memory. It performs this check every 600 seconds (the default). For every process it finds exceeding the limit, it will end out exactly one notification. In addition to the normal `hrSWRunPerfMem` oid and value sent in the trap, the `hrSWRunName` object will also be sent.

Note that the `hrSWRunName` object actually occurs in a different table, but since the indexes to the two tables are the same this works out all right. indexes to the two tables are the same this has no effect.

```
rouser admin
monitor -u me -o sysUpTime.0 -o hrSWRunName "high process memory"
hrSWRunPerfMem > 10000
```

The above line would produce a trap which is formatted by snmptrapd as follows:

```
2002-04-05 13:33:53 localhost.localdomain [udp:127.0.0.1:32931]:
sysUpTimeInstance = Timeticks: (1629) 0:00:16.29
snmpTrapOID.0 = OID: mteTriggerFired mteHotTrigger = high process memory
mteHotTargetName = mteHotContextName = mteHotOID = OID: hrSWRunPerfMem.1968
mteHotValue = 28564 hrSWRunName.1968 = "fw"
```

This shows the `fw` process using 28Mb of resident memory.

### **defaultMonitors yes**

By default, the agent and the DISMAN-EVENT-MIB support do nothing until configured. Typically users wish to watch a number of tables within the UCD-SNMP-MIB, which are designed specifically for reporting problems.

If the “defaultMonitors yes” line is present in the `snmpd.conf` file (which must be accompanied by an appropriate `agentSecName` line and a `rouser` line), the following monitoring conditions will be installed:

```
monitor -o prNames -o prErrMsg "process table" prErrorFlag != 0
monitor -o memErrorName -o memSwapErrorMsg "memory" memSwapError != 0
monitor -o extNames -o extOutput "extTable" extResult != 0
monitor -o dskPath -o dskErrorMsg "dskTable" dskErrorFlag != 0
monitor -o laNames -o laErrMsg "laTable" laErrorFlag != 0
monitor -o fileName -o fileErrorMsg "fileTable" fileErrorFlag != 0
```