The Sage

The Sage is a portable software package designed to enable a network manager to request SNMP variables from nodes in the network that use SNMP as network management protocol. Examples of such nodes are The Beholder, "the" network monitor, CISCO routers and UNIX hosts.

With The Sage, a number of useful applications can be created. Using a number of Beholders in the network, a simple Sage program can continuously report the network load in all segments. An other Sage program could request performance data on fixed time intervals from the Beholders, and dump them in log files. These files could easily be transformed in management reports using a program like SAS, Lotus 123 or plot.

The core of The Sage consists of a SCHEME interpreter. SCHEME is a modern programming language which is best described as a mix of LISP and C. This interpreter is enhanced with a set of SCHEME functions that make up a full fledged SNMP library. By writing SCHEME functions that use this SNMP lib, SNMP nodes in the network can be reached, and the data can be handled in some meaningful way.

The Sage is written in standard C, and can be compiled by both ANSI C compilers and standard UNIX C compilers. The SNMP library uses the TCP/IP socket library to communicate with the network. The code of The Sage is highly portable and has been used on UNIX, OS/2 and MSDOS systems.

The reader of this user manual is supposed to have knowledge of the SCHEME programming language.

Installation

The Sage is distributed in source code, so that it can be recompiled on practically any environment you would like. Remember, we made this stuff, so don't reuse the software without giving credit where credit is due.

Installing The Sage consists of four steps:

. unpack the Sage.tar.Z file in your own development tree using something like:

```
% cd /users/netmgr
% uncompress Sage.tar.Z
% tar -xvf Sage.tar
```

. edit Makefile to set the variables SAGEPATH and BINPATH correctly
Good values are:

```
SAGEPATH = /usr/local/lib/Sage
BINPATH = /usr/local/bin
```

. run "make all"

. run "make install"

The Sage SNMP library

The Sage SNMP library was designed to enable the programmer to access all data stored in the network. It is possible to work with communities, multiple concurrent SNMP sessions and variable time-outs.
The following describes every function available in the library.

SNMP-OPEN

Synopsis

(SNMP_OPEN *host port*) -> *connection_number*

host string containing host name
port UDP port number (normally 161)

Description

The SNMP_OPEN call opens a UDP connection to a SNMP agent. The returned connection number can be used to feed the handling of SNMP variables in the SNMP host.

Returns

NIL if error, else connection number

SNMP-CLOSE

Synopsis

(SNMP_OPEN *connection_number*)

connection_number number of connection to close

Description

The SNMP_CLOSE call closes a UDP connection with a SNMP agent.

Returns

none

SNMP-GET

Synopsis

(SNMP_GET *conn community timeout retreis var_list*) -> *var_list*

conn connection number
community string containing community of request
timeout timeout of each try in msecs
retreis number of time the request should be retried
var_list list with variables to be requested. The list should have the form:
((*var-id var-type var-val*) .....)

var-id string with ASN.1 id, dotted format
var-type integer with ASN.1 variable type
var-val SCHEME value of the variable

Description

The SNMP_GET does a GET of the list of SNMP variables given in the variable <var_list>. The values given in the variables will not be used. The resulting var_list can be used in a GET_NEXT request.

Returns

list with SNMP variables, NIL if error.

SNMP-GETNEXT

Synopsis

(SNMP_GETNEXT *conn community timeout retreis var_list*) -> *var_list*

conn connection number
community string containing community of request
timeout timeout of each try in msecs
retreis number of time the request should be retried
var_list list with variables to be requested. The list should have the form:
((*var-id var-type var-val*) .....)

var-id string with ASN.1 id, dotted format

var-type integer with ASN.1 variable type
var-val SCHEME value of the variable

Description

The SNMP_GETNEXT does a GETNEXTof the list of SNMP variables given in the variable <var_list>. The values given in the variables will not be used. The resulting var_list can be used in a following GETNEXT request.

Returns

list with SNMP their values, NIL if error.

SNMP-SET

Synopsis

(SNMP_SET *conn community timeout retreis var_list*) -> *var_list*

conn	connection number
community	string containing community of request
timeout	timeout of each try in msecs
retreis	number of time the request should be retried
var_list	list with variables to be requested. The list should have the form: ((*var-id var-type var-val*) .....)

var-id string with ASN.1 id, dotted format
var-type integer with ASN.1 variable type
var-val SCHEME value of the variable

Description

The SNMP_SET does a SET of the list of SNMP variables given in the variable <var_list>. The resulting var_list can be used in a GET_NEXT or GET request.

Returns

list with SNMP variables, NIL if error.

The Sage Utility library

SLEEP

Synopsis

(SLEEP *sleeping_time*) -> *void*

sleeping_time time in msecs to sleep

Description

The function lets the Sage sleep for a while.

Returns

void

GETENV

Synopsis

(GETENV *environment_name*) -> *environment_value*

environment_name name of environment variable

Description

This function returns the string containing the value of a environment variable.

Returns

value environment variable

The Sage Curses library

This library is being build, and will give Sage programs the ability to display windows with information on a standard UNIX display.

Example

SNMP seems like a very daunting subject. It involves complex networks and even more complex network information coded in some weird OSI protocol. Most people needing the information are not willing to dig in the network and get the information out.
Sage helps by simplifying the collection phase. The following Sage program retrieves a SNMP variable from a SNMP host in the network.

```
<1> (define connection
        (SNMP-OPEN "beholder1.et.tudelft.nl" 161))
<2> (define result
        (SNMP-GET connection "public" 6000000 3 '(("1.2.3.4.5.6.7" 0 0) ("2.3.4.5.6.7" 0 0)))
<3> (SNMP-CLOSE connection)
<4>(print "system description = ") (print (caddr result)))
```

Line <1> opens a connection with the well known host beholder1.et.tudelft.nl. The host should be in the hosttable, or, if you have a good nameserver, a full hostname like the example. The UDP port number 161 is standard for SNMP agents. Use an other value only in test environments.

Line <2> retrieves the value of the SNMP variable that gives the system description. The number is defined in the standard SNMP MIB. The variable type and variable value are ignored because the is a GET request. The result stored in the variable <result> has the same format as the input variable definition, with the type and value replaced by their current values.

Line <3> closes the connection again.

Line <4> does something meaningful with the results. Note that normally you will know what type of variable you are retrieving, so you know how to handle the returned list of variables. The example doesn't check for error conditions.