NAME

snmpd.ea - extensible daemon that responds to SNMP requests

SYNOPSIS

snmpd.ea [**-C** *contact*] [**-L** *location*] [**-P** *port*] [**-a**] [**-c**] [**-e** *extendfile*] [**-k**] [**-l** *logfile*] [**-m** *logmask*] [**-p**]

snmpd.ea [**-M** *logmask*]

DESCRIPTION

snmpd.ea is an agent that enables you to extend the agent's existing Management Information Base (MIB) to support new MIB objects. The manager sends requests for MIB values to the agent using the Simple Network Management Protocol (SNMP). The agent replies with the information requested.

The extensible agent, **snmpd.ea,** is a superset of the *snmpd (1M).* The **snmpd.ea** responds to the same objects as the **snmpd** and can also be configured to respond to additional MIB objects.

The MIB is a conceptual database of values on the agent. See **/usr/OV/snmp_mibs/hp-unix** for a list of MIB values that **snmpd.ea** supports. The agent supports all objects in RFC 1213 except the EGP group.

Only the super-user can start **snmpd.ea**, and only one **snmpd.ea** can execute at a time.

When invoked, **snmpd.ea** reads **/etc/snmpd.conf** and **/etc/snmpd.extend** to configure itself (see *snmpd.conf*(4) and *snmpd.extend*(4)).

Options

snmpd.ea recognizes the following options:

-C *contact* Specify the contact person responsible for the network management agent. This option overrides the contact person specified in **/etc/snmpd.conf**.

-L *location* Specify the location of the agent. This option overrides the location specified in **/etc/snmpd.conf**.

-M *logmask* Send a message to the currently running **snmpd.ea** to change its logging mask to *logmask*. See the Log Masks section for valid values.

-P *port* Specify the UDP port number the agent will listen on for requests.

-a Suppress sending authenticationFailure traps.

-c Reconfigure **snmpd.ea** (force **snmpd.ea** to re-read **/etc/snmpd.conf** and **/etc/snmpd.extend)**.

-e *extendfile* Use *extendfile* for adding objects defined in a different file than the default extend file, **/etc/snmpd.extend**.

-k Kill the currently running **snmpd.ea**.

-l *logfile* Use *logfile* for logging rather than the default logfile, **/usr/adm/snmpd.log**.

-m *logmask* Sets the initial logging mask to *logmask*. See the Log Masks section for valid values.

-p Parse the extend file and exit. This is used to check the extend file for syntax errors.

Traps

The agent also sends information to a manager without an explicit request from the manager. Such an operation is called a "trap." **snmpd.ea** sends the following SNMP traps:

coldStart Sends a coldStart trap when **snmpd.ea** is invoked.

linkDown Sends a linkDown trap when an interface goes down.

linkUp Sends a linkUp trap when an interface comes up.

authenticationFailure
Sends an authenticationFailure trap when an SNMP request is sent to **snmpd.ea** with a community name that does not match the community names specified in **/etc/snmpd.conf**.

Each SNMP request is accompanied by a community name, which is a password that enables SNMP access to MIB values on an agent. A manager can request to read a MIB value by issuing an SNMP GetRequest, or a manager may request to alter a MIB value by issuing an SNMP SetRequest.

Log Masks

Log masks specify the type of output listed in **/usr/adm/snmpd.log** or in *logfile*. To select multiple output types, add the individual *logmask* values together and enter that number.

0 Turn off logging.

1 Log authenticationFailure traps.

2 Log errors.

4 Log configuration requests.

8 Log requests and replies.

16 Log requests and replies for objects that have been added.

32 Log hexdumps of packets received and sent by **snmpd.ea**.

64 Log trace messages.

Defaults

By default, the agent (**/etc/snmpd.ea**) does not allow managers to alter MIB values (it returns errors for SNMP SetRequests). To configure the agent to respond to SNMP SetRequests, add a **set-community-name** to **/etc/snmpd.conf**.

By default, the agent responds to all SNMP GetRequests, regardless of community name used in the request. To configure the agent to respond to a specific community name, add a **get-community-name** to **/etc/snmpd.conf**.

By default, SNMP traps are not sent to any destination. To configure the agent to send traps to one or more specific destinations, add the trap destinations to **/etc/snmpd.conf**.

By default, the agent's location is a blank string. To configure the agent's location, add the location to **/etc/snmpd.conf**, or use the **-L** option.

By default, the agent's contact is a blank string. To configure the agent's contact, add the contact to **/etc/snmpd.conf**, or use the **-C** option.

By default, the agent only responds to objects defined in RFC 1213 and the HP-UNIX MIB. When a file **/etc/snmpd.extend** exists, the agent responds to those objects defined in **/etc/snmpd.extend.**

By default the agent logs authenticationFailure traps and errors to **/usr/adm/snmpd.log** or in *logfile*.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages appear. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **snmpd.ea** behaves as if all internationalization variables are set to "C." See *environ*(5).

International Code Set Support

Supports single-byte character code sets.

AUTHOR

snmpd.ea was developed by HP and Massachusetts Institute of Technology.

FILES

/etc/snmpd.conf
/usr/adm/snmpd.log
/etc/snmpd.extend
/usr/OV/snmp_mibs/hp-unix

SEE ALSO

chksnmpd(1), snmpd.conf(4), snmpd.extend(4).

RFC 1155, RFC 1157, RFC 1212, RFC 1213.

NAME

chksnmpd - check connectivity with the SNMP agent

SYNOPSIS

chksnmpd [**-x**] [**-v**]

DESCRIPTION

chksnmpd performs a loop-back, SNMP request to the SNMP agent to get a MIB value. The request is sent to either the *snmpd*(1M) or the *snmpd.ea*(1M). If **chksnmpd** receives a valid response, it reports a "Success" message; otherwise, it reports a "Failure" message.

If there is a community name entered in **/usr/OV/conf/ovsnmp.conf**, **chksnmpd** uses that community name; otherwise, **chksnmpd** uses the **public** default community name.

chksnmpd checks connectivity with the local SNMP agent executing on the same system that the **chksnmpd** command is executing. **chksnmpd** can not check connectivity with an SNMP agent running on a remote system.

Options

-x Hexdump the packets sent to the SNMP agent and the packets received from the SNMP agent.

-v Display the results of the response.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages appear. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **chksnmpd** behaves as if all internationalization variables are set to "C." See *environ*(5).

International Code Set Support

Supports single-byte character code sets.

AUTHOR

chksnmpd was developed by HP.

FILES

/etc/snmpd.conf

SEE ALSO

snmpd(1M), snmpd.ea(1M), snmpd.conf(4).

RFC 1155, RFC 1157, RFC 1212, RFC 1213.

NAME

snmptrap - issue an SNMP Trap

SYNOPSIS

snmptrap *[options] node enterprise agent-addr generic-trap specific-trap time-stamp* [*variable type value ...*]

DESCRIPTION

snmptrap issues an SNMP Trap to *node*. *node* can be either an Internet address or a host name (see *hosts*(4)).

enterprise has the format of .A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation. An *enterprise* of "" (empty string) is interpreted as the default object identifier as appropriate, based on the hardware in use.

agent-addr must be either an Internet address or a host name (see *hosts*(4)). An *agent-addr* of "" (empty string) is interpreted as the local hostname.

generic-trap must be a whole number between zero and six (inclusive).

specific-trap can be any 32-bit integer.

time-stamp must be a whole number equal to or greater than zero.

Each *variable* has the format of .A.B.C.D..., where A, B, C, and D are subidentifiers in decimal notation.

Each *type* must be one of the following types: integer octetstring octetstringhex octetstringoctal octetstringascii objectidentifier null ipaddress counter gauge timeticks opaque opaquehex opaqueoctal opaqueascii.
See RFC 1155 for a complete description of each *type*.

The *value* must be valid for the *type* specified. When using a *type* where a hexadecimal or octal *value* is needed, the *value* must have each byte fully defined. For example, **fff** (or **17377**) is not allowed, whereas **0fff** (or **017377**) is. For *type* Null, a *value* must be specified on the command line, but it is ignored when the request is created. *value* must occupy not more than 512 bytes maximum.

Options

-d Dump (list) input and output packets in a hexidecimal and partially ASN.1 decoded format.

-p Override the default port for sending/receiving to *port*.

-c Override the SNMP community name (as configured in **/usr/OV/conf/ovsnmp.conf**) to *community*. See *ovsnmp.conf*(4).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages appear. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, *snmptrap* behaves as if all internationalization variables are set to "C." See *environ*(5).

International Code Set Support

Supports single-byte character code sets.

EXAMPLES

The following command sends an SNMP trap to node *testnode* setting the agent and enterprise to the machine upon which the command is run and with *time-stamp* equal to 15. The trap has *generic-trap* equal to 6, and *specific-trap* equal to 15. The information passed with the trap is **system.sysDescr.0**.

```
snmptrap testnode "" "" 6 1 15 system.sysDescr.0 \
        octetstringascii "SunOS hpcndnw2 4.1 1 sun4c"
```

AUTHOR

snmptrap was developed by Carnegie-Mellon University and HP.

SEE ALSO

snmpd(1M), trapd(1M), trapd.conf(4)

/usr/OV/doc/* (for related RFC's)

NAME

snmpd.conf - configuration file for the SNMP agent

DESCRIPTION

When invoked, the SNMP agent reads its configuration information from the **/etc/snmpd.conf** configuration file. The SNMP agent is either the *snmpd*(1M) or the *snmpd.ea*(1M). The SNMP agent operates correctly if no values are configured in **/etc/snmpd.conf.**

/etc/snmpd.conf contains the following configurable values:

get-community-name:
Specifies community name for the agent. The agent responds to SNMP GetRequests with this community name. You can configure the agent to respond to more than one get community name. If a community name is not entered, the agent responds to SNMP GetRequests using any community name.

set-community-name:
Specifies community name for the agent. The agent responds to the SNMP SetRequests with this community name. You can configure the agent to respond to more than one set community name. If a community name is not entered, the agent returns an error.

trap-dest: Specifies the system name where traps are sent (that is, the trap destination). This system name is usually the host name or IP address of the manager.

location: Specifies the physical location of the agent.

contact: Specifies the person responsible for this agent and information on how to contact this person.

Separate the fields by blanks or tabs. A **#** character indicates the beginning of a comment; characters from the **#** character to the end of the line are ignored.

EXAMPLES

Each line in the following example **snmpd.conf** file is preceded by a comment (beginning with **#)** that explains the entry.

```
# Restrict the agent to responding
# only to SNMP GetRequests that
# have the community name secret

get-community-name: secret

# Allow the agent to respond
# to SNMP SetRequests with
# either the community name private
# or secret

set-community-name: private
set-community-name: secret

# Allow the agent to respond
# to SNMP SetRequests that
# have the community name private

set-community-name: private

# Send traps to system names
# manager1 and 15.2.113.233

trap-dest: manager1
trap-dest: 15.2.113.233

# Specify the agent is located
# on the first floor
# near the mens room

location: 1st Floor near Mens Room
```

```
# Specify Bob Jones is responsible
# for this agent and his
# phone number is 555-2000

contact: Bob Jones (Phone 555-2000)
```

AUTHOR

snmpd.conf was developed by HP.

SEE ALSO

chksnmpd(1), snmpd(1M), snmpd.ea(1M).

RFC 1155, RFC 1157, RFC 1212, RFC 1213

NAME

snmpd.extend - configuration file containing the snmpd.ea MIB definitions

DESCRIPTION

When invoked, *snmpd.ea*(1M) reads its configuration information from the **/etc/snmpd.extend** configuration file. *snmpd.ea*(1M) operates correctly if no values are configured in **snmpd.extend**.

The **snmpd.extend** file is the MIB module that extends the MIB on the agent to include the objects you define. The **snmpd.extend** file is designed to use the macro template defined in *Concise MIB Definitions*, RFC 1212. Therefore, when you create the **snmpd.extend** file, follow the Abstract Syntax Notation One (ASN.1) format described in RFC 1212. Use the Hewlett-Packard enterprise-specific MIB or the Internet-standard MIB-II as a model. You can access these MIBs online. The respective files are **/usr/OV/snmp_mibs/hp-unix** and **/usr/OV/snmp_mibs/rfc1213-MIB-II**

The MIB objects you define may be of two types. The objects may be associated with commands or they may be associated with a file. If the object is associated with a command and the agent receives an SNMP request for that object, the command is executed and the output of the command is returned in the SNMP reply. If the object is associated with a file and the agent receives an SNMP request for that object, the file is read and the contents of the file are returned in the SNMP reply.

When choosing whether to define objects as command objects or file objects you may want to ask the following questions:

Is a command required to obtain the object's value? If so, you may want to define the object associated with a command. If the value of the object doesn't require a command to be executed every time the object's value is retrieved, you may want to define the object associated with a file.

Are the objects arranged in a table? Do the objects have rows and columns? If so, you must define the objects associated with a file.

Are you concerned with the performance of executing a command each time the object is retrieved? If so, you may want to avoid executing a command and define the objects associated with a file.

The command-based MIB objects must have commands associated with the object. The command can be either existing UNIX commands, or commands that you have written. You specify these commands in the **DESCRIPTION** field in the MIB module. See the "Commands" section for details on writing commands.

The file-based MIB objects must have a file name associated with the object. You specify the file name in the **DESCRIPTION** field in the MIB module. See the "Files" section for details on writing files.

MACRO

Note that the **snmpd.extend** file differs from the RFCs in the following areas:

IMPORTS is not required and will be ignored if added.

EXPORTS is not required and will be ignored if added.

DESCRIPTION is required.

When defining objects associated with commands, note that the **snmpd.extend** file differs from the RFCs in the following areas:

SYNTAX cannot be SEQUENCE OF

INDEX is not supported. The **INDEX** caluse is supported for file-based MIB objects. See the "Files" section for details.

This is the macro template for **/etc/snmpd.extend**. The various fields are described later in this entry.

```
moduleName DEFINITIONS ::= BEGIN

-- dashes indicate a comment

enterpriseName      OBJECT IDENTIFIER ::= { objectID }
nodeName      OBJECT IDENTIFIER ::= { objectID }

Object OBJECT-TYPE

    SYNTAX Value
    ACCESS Value
    STATUS Value
```

DESCRIPTION

"**Add a textual description of your object here.**
READ-COMMAND: read_command
READ-COMMAND-TIMEOUT: timeout_in_seconds
WRITE-COMMAND: write_command
WRITE-COMMAND-TIMEOUT: timeout_in_seconds
FILE-COMMAND: file_command"
FILE-COMMAND-FREQUENCY: file_command_seconds
PIPE-IN-NAME: pipe_in_name
PIPE-OUT-NAME: pipe_out_name
PIPE-FREQUENCY: pipe_seconds
APPEND-COMMUNITY-NAME: true | false
FILE-NAME: file_name"

::= { parent_node subidentifier }

END

snmpd.extend contains the following configuration values:

moduleName Is the name of your MIB module.

–– To add documentation, insert two dashes (––) in front of comments.

enterpriseName
Is the enterprise ID you have registered with the Internet Assigned Numbers Authority. For example, Hewlett-Packard's enterprise ID is hp and the corresponding objectID is { enterprises 11 }.

nodeName Is the name of the node under which you want to organize your MIB objects. A node can be a child of another node.

Object Is the textual label of your object.

SYNTAX Defines the data structure corresponding to the object type. The HP OpenView Extensible SNMP Agent supports all data structures defined in the "Syntax" section below.

ACCESS Defines whether an object's value can be read and written. See the "Access" section for valid values.

STATUS Defines the implementation support required. Valid values for **STATUS** are **mandatory**, **optional**, **obsolete**, and **deprecated**. Always use **mandatory**.

DESCRIPTION Describes your object. This is also where you either define the commands associated with the object or define the file associated with the objects. The object must be associated with an object or a file, the object can not be associated with both. See the "Description" section for valid values.

Syntax
The **SYNTAX** clause can have the following values:

INTEGER A simple type consisting of positive and negative whole numbers, including zero. Do not use zero as an enumerated type.

OCTET STRING A simple type taking zero or more octets, each octet being an ordered sequence of eight bits.

OBJECT IDENTIFIER
A type denoting an authoritatively named object. An example would be 1.3.6.1.2.1.1.

NULL A simple type consisting of a single value, also called null. This type can only be used with defining an object associated with a command.

NetworkAddress A type representing an IP address.

IpAddress A type representing an IP address.

Counter A type representing a non-negative integer which monotonically increases until it reaches a maximum value, when it wraps around and starts increasing again

from zero.

Gauge A type representing a non-negative integer, which may increase or decrease, but which latches at a maximum value.

TimeTicks A type representing a numeric value which counts the time since an event.

Opaque A type representing an arbitrary encoding.

DisplayString A type representing textual information taken from the NVT ASCII character set as defined in pages 10-11 of RFC 854.

PhysAddress A type representing a media address. For many types of media, this will be in a binary representation. For example, an ethernet address would be represented as a string of 6 octets.

SEQUENCE OF A type representing a table. This type represents objects that have rows and columns. This type can only be used when defining objects associated with a file.

SEQUENCE A type representing the entry of a table. This type is a child of an object with type **SEQUENCE OF.** This type may optionally have an **INDEX** clause. The **INDEX** clause identifies the column that uniquely defines the row. The default **INDEX** clause is the first column of the table. This type can only be used when defining objects associated with a file.

The maximum value for counters, gauges, integers, and timeticks is 2^32-1 (4 294 967 295 decimal.)

Access

The **ACCESS** clause can have the following values:

read-only can read values but can not write values

read-write can read and write values

write-only can write values but can not read values

Description

The **DESCRIPTION** clause is where you put you textual description.

If your object is associated with a command, enter the following fields after the textual description:

READ-COMMAND If your **ACCESS** value is **read-only** or **read-write**, the **READ-COMMAND** must be entered. When the agent receives an SNMP GetRequest or GetNextRequest, the **snmpd.ea** executes the **read_command** and returns the output of that command in the SNMP Reply. The output can be either the standard output or the standard error of the command.

WRITE-COMMAND If your **ACCESS** Value is **read-write** or **write-only**, the **WRITE-COMMAND** must be entered. When the agent receives an SNMP SetRequest, the **snmpd.ea** executes the **write_command**. The **write_command** should not generate output to standard error or standard out.

READ-COMMAND-TIMEOUT
specifies the time in seconds the agent should wait for **read_command** to finish.

WRITE-COMMAND-TIMEOUT
specifies the time in seconds the agent should wait for **write_command** to finish. If the **timeout_in_seconds** is –1, the **write_command** is executed and the agent responds without waiting for the command to finish.

Both **READ-COMMAND-TIMEOUT** and **WRITE-COMMAND-TIMEOUT** are optional. If the commands do not return before **timeout_in_seconds,** the agent kills **read_command/write_command** and returns a general error. The default **timeout_in_seconds** is 3 seconds and the maximum timeout value is 30 seconds. The **DESCRIPTION** clause may contain the **READ-COMMAND, READ-TIMEOUT, WRITE-COMMAND,** and **WRITE-TIMEOUT** fields if the **SYNTAX** is one of the following: **INTEGER**, **OCTET STRING**, **OBJECT IDENTIFIER**, **NULL**, **NetworkAddress**, **IpAddress**, **Counter**, **Gauge**, **TimeTicks**, **Opaque**, **DisplayString**, or **PhysAddress**.

If your object is associated with a file, enter the following fields after the textual description.

FILE-COMMAND When the agent receives an SNMP GetRequest, GetNextRequest, or SetRequest, the **snmpd.ea** executes **file_command** before either reading or creating **file_name.** When the agent receives an SNMP SetRequest, the **snmpd.ea** also executes the **file_command** after the **file_name** has been created. The **file_command** must complete within 10 seconds.

FILE-COMMAND-FREQUENCY
When the agent receives an SNMP GetRequest or GetNextRequest, the **snmpd.ea** executes **file_command** if the agent last executed **file_command** more than **file_command_seconds** ago. By default, the **file_command** will get executed at most every 10 seconds. The agent executes **file_command** both before and after the file has been created for every SNMP SetRequest regardless of when it was last executed.

PIPE-OUT-NAME When the agent receives an SNMP GetRequest, GetNextRequest, or SetRequest, the **snmpd.ea** writes to **pipe_out_name .** The **pipe_out_name** must be a FIFO (named pipe). The management station's IP address, the community name used in the request, the OBJECT IDENTIFIER used in the request, the SYNTAX of the object, the request issued by the management station, and the instance are written to the pipe. Each value is separated by white space and the message ends with the '\0' character. When the agent receives an SNMP SetRequest, the **snmpd.ea** writes to **pipe_out_name** before and after the **file_name** has been created.

PIPE-IN-NAME After the agent writes to **pipe_out_name,** the **snmpd.ea** reads **pipe_in_name** waiting for a message. If the agent reads a 0, the agent continues processing and either reads or creates **file_name.** If the agent reads something other than 0, or if the agent doesn't receive a message within 10 seconds, the agent returns an genErr. The **PIPE-IN-NAME** is required if the **PIPE-OUT-NAME** is present. Both **pipe_in_name** and **pipe_out_name** can be created used the **mkfifo(1)** command.

PIPE-FREQUENCY When the agent receives an SNMP GetRequest or GetNextRequest, the **snmpd.ea** writes to **pipe_out_name** if the agent last wrote to **pipe_out_name** more than **pipe_seconds** ago. By default, the **pipe_out_name** will get written to at most every 10 seconds. The agent writes to **pipe_out_name** both before and after the file has been created for every SNMP SetRequest regardless of when it was last written to.

APPEND-COMMUNITY-NAME
If **APPEND-COMMUNITY-NAME** is **true**, the **snmpd.ea** reads or creates **file_name.comm** where **comm** is the community name sent in the request. If **file_name.comm** is not present, the agent returns an error. The value for **APPEND-COMMUNITY-NAME** must be either **true** or **false**.

FILE-NAME When the agent receives an SNMP GetRequest or GetNextRequest, the **snmpd.ea** reads the **file_name** and returns the contents of that file in the SNMP Reply. When the agent receives an SNMP SetRequest, the **snmpd.ea** creates **file_name** containing the value that the object is set to. The **file_name** will no longer contain comments after the **snmpd.ea** creates it.

The **FILE-COMMAND**, **FILE-COMMAND-FREQUENCY**, **PIPE-IN-NAME**, **PIPE-OUT-NAME**, **PIPE-FREQUENCY**, and **APPEND-COMMUNITY-NAME** clauses are optional. The **DESCRIPTION** clause may contain the **FILE-NAME** field if the **SYNTAX** is one of the following: **INTEGER**, **OCTET STRING**, **OBJECT IDENTIFIER**, **NULL**, **NetworkAddress**, **IpAddress**, **Counter**, **Gauge**, **TimeTicks**, **Opaque**, **DisplayString**, **PhysAddress**, or **SEQUENCE OF.**

After the **snmpd.ea** initially reads **file_name** it will only re-read it if the file has been modified.

Commands
The **read_command** and **write_command** and **file_command** are executed as if executed by **/bin/sh**. By default, **snmpd.ea** does not pass any arguments to the **read_command** or **file_command**. By

default, one argument is passed to the **write_command** — the value that the object is set to.

The following notation is used to pass arguments to the **snmpd.ea**. The arguments can be specified in any order. The arguments are passed as strings.

- **$i** The management station's IP address. The address is in internet dot notation.
- **$c** The community name used in the request.
- **$o** The OBJECT IDENTIFIER used in the request. The OBJECT IDENTIFIER is in dot notation.
- **$s** The SYNTAX of the object. One of the following values will be passed: **ObjectIdentifier**, **OctetString**, **Integer**, **NetworkAddress**, **IpAddress**, **Counter**, **Gauge**, **TimeTicks**, **Opaque**, **Null**, **DisplayString**, or **PhysAddress**.
- **$r** The request issued by the management station. One of the following values will be passed: **GetRequest**, **GetNextRequest**, **SetRequest**, or **PostSetRequest**. The **PostSetRequest** will be passed to the **file_command** after the **file_name** has been created.
- **$I** The instance used in the request.
- **$*** All arguments. The same as **$i $c $o $s $r $I**.
- **$$** Substitute $.

If the **snmpd.extend** contains **READ-COMMAND: /usr/local/bin/eagent_read $i $c $o $s $I** and an SNMP GetRequest comes from a management station with IpAddress 16.17.18.19, community name public, for Object Identifier 1.3.6.1.2.3.4.5.6.7, with syntax Gauge, with instance 0, the agent would execute "/usr/local/bin/eagent_read 16.17.18.19 public 1.3.6.1.2.3.4.5.6.7 Gauge 0."

The command should exit with a valid exit code. Any invalid exit codes result in a general error returned to the management station. A **read_command** should exit with

- **0** No error. The command was successful. The data echoed to standard output or standard error will be returned to the management station in the SNMP reply. Only data necessary for the reply should be echoed to standard output or standard error. Too much data returns a general error. The data echoed must be of the same **SYNTAX** as specified in **snmpd.extend.** If the data is not of the same **SYNTAX,** a general error is returned to the management station.
- **5** General error. The command was unsuccessful.

A **write_command** should exit with

- **0** No error. The command was successful. No data should be echoed to standard output or standard error.
- **3** Bad value. The value passed into the command was invalid. For example, if the command accepts values from 1-9, and the value 132 is passed into the command, the command should exit with 3.
- **5** General error. The command was unsuccessful.

A **file_command** should exit with 0. A general error is returned if the **file_command** doesn't exit with 0.

Files

The **file_name** contains the value of the object. For objects that have **SYNTAX** other than **SEQUENCE OF** the file must contain the value of the object. If the contents of the file is not of the same SYNTAX, a general error is returned to the management station.

If the object has **SYNTAX SEQUENCES OF**, the file contents represent a table. Each row of the table is separated by a new-line and each row contains a column for each object in the table. A row can continue over a new-line by adding the character '\' at the end of the line. A column's value may not extend over a new-line.

The row in the table is uniquely defined by the object[s] defined in the **INDEX** clause. If the **INDEX** clause is not present, the **snmpd.ea** assumes that the first column of the file uniquely defines the row. The value[s] for the objects[s] defined to be the **INDEX** of the table must all be unique.

Values entered in **file_name** must be separated by white space (spaces, tabs, ...). The value may be enclosed in double-quotes. If a value contains a double quote, the quote can be embedded using \". For example, a row in a file may contain the following four values:

**"column \"in\" quotes " column2 3 **
column4

Each of the four values represents a column within a table. The first column is a string with embedded quotes. The fourth column is a string and is located on the second line of the file.

A **#** character in the first column indicates a comment.

When creating **file_name** it is recommended that a temporary file be created and then placed in the correct location after the temporary file has been completed. Use mv(1) or cp(1) to place the temporary file in the correct location.

EXAMPLES

An example **snmpd.extend** can be found in **/usr/OV/prg_samples/eagent/snmpd.extend**

Example commands can be found in **/usr/OV/prg_samples/eagent/num_widgets** and **/usr/OV/prg_samples/eagent/change_num_widget**

Example files can be found in **/usr/OV/prg_samples/eagent/user_disk_space** and **/usr/OV/prg_samples/eagent/root_processes**

Example file commands can be found in **/usr/OV/prg_samples/eagent/get_processes** and **/usr/OV/prg_samples/eagent/update_inetd.**

AUTHOR

snmpd.extend was developed by HP.

SEE ALSO

chksnmpd(1), mkfifo(1), snmpd.ea(1M), snmpd.conf(4).

RFC 1155, RFC 1157, RFC 1212, RFC 1213.