



Product:	SNMPManager
Product version:	V2.7
Document ID:	UM-SNMPManager
Doc revision:	A
Written/Aprr.:	OH / SL
Date:	14. Nov. 2008

Industrial Control Design AS



SNMPManager V2.7

User Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. INTRODUCTION.....	4
1.1. About.....	4
1.2. Terms and Definitions.....	4
2. CONFIGURATION.....	6
2.1. NetworkInterface.....	6
2.1.1. Description.....	6
2.1.2. Example XML.....	6
2.1.3. Elements.....	6
2.1.4. Required modifications.....	6
2.2. Signals.....	6
2.2.1. Description.....	6
2.2.2. Example XML.....	6
2.2.3. Elements.....	6
2.2.4. Required modifications.....	6
2.3. Alarms.....	7
2.3.1. Description.....	7
2.3.2. Example XML.....	7
2.3.3. Elements.....	7
2.3.4. Required modifications.....	7
2.4. Parameters.....	7
2.4.1. Description.....	7
2.4.2. Example XML.....	7
2.4.3. Elements.....	7
2.4.4. Required modifications.....	7
2.5. SNMPAgents.....	7
2.5.1. Description.....	7
2.5.2. Example XML.....	7
2.5.3. Elements.....	8
2.5.4. Required modifications.....	8
2.6. ReceiveTraps.....	8
2.6.1. Description.....	8
2.6.2. Example XML.....	8
2.6.3. Elements.....	8
2.6.4. Required modifications.....	9
2.7. SetRequests.....	9
2.7.1. Description.....	9
2.7.2. Example XML.....	9
2.7.3. Elements.....	10
2.7.4. Required modifications.....	10
2.7.5. Supported Types in SetReq.....	10
2.8. GetRequests.....	10
2.8.1. Description.....	10
2.8.2. Example XML.....	10
2.8.3. Elements.....	10
2.8.4. Required modifications.....	11
2.8.5. Supported Types in GetReq.....	11
2.9. PollIntervals.....	11
2.9.1. Description.....	11
2.9.2. Example XML.....	11

2.9.3. Elements.....	11
2.9.4. Required modifications.....	11
2.10. NumbIntervForTimeout.....	12
2.10.1. Description.....	12
2.10.2. Example XML.....	12
2.10.3. Elements.....	12
2.10.4. Required modifications.....	12
2.11. List of necessary modifications in XML-file.....	12
2.12. SNMP Agent Setup.....	12

3. FUNCTIONAL DESCRIPTION.....14

3.1. Set Requests.....	14
3.1.1. SetSNMPvalue-message.....	14
3.2. Get Requests.....	14
3.2.1. GetSNMPvalue-message.....	14
3.2.2. GetSNMPvalueReply.....	15
3.3. Traps.....	15
3.3.1. Functionality for recognized Traps.....	15
3.3.2. Unknown SNMP Agent Alarm.....	16
3.3.3. Supported value types (identifier).....	17

4. APPENDIX.....18

4.1. Example SNMPManager Component XML-file.....	18
4.2. SNMP Protocol.....	21
4.2.1. Introduction.....	21
4.2.2. Format.....	21

1. Introduction

1.1. About

This document describes how the SNMP Manager CDP component works, and how to set it up and use it with the CDP system. The SNMP Manager CDP component has the following features:

- The SNMP Manager is able to send SNMP Set and Get requests to several agents (SNMP v. 1 and 2).
- Get requests can be sent regularly, specified by two different intervals.
- Responses from agents can be used to update CDP Signals.
- It can receive two external messages, SetSNMPvalue and GetSNMPvalue, and send back an external message GetSNMPvalueReply.
- It is able to receive SNMP Trap messages sent from several sources (SNMP v. 1 and 2).
- It is able to decode the received Trap messages, and generate and Set readable Alarm-messages with updated Description-field based on the received trap. These Alarm-messages can be read in a standard web-browser or in the CDP Browser. The messages can also be logged.
- The received traps can also be specified to either Set or Clear a unique Alarm. Such unique Alarms are auto-created based on specified Name, AgentNumbers and optional Values. They can also optional be routed to CDP Signals by SignalOutSet.
- Configuration of the component is done in the component's XML-file (SNMPManager.xml).

1.2. Terms and Definitions

Agent

Software that enables a device to respond to manager requests to view or update MIB data, and send traps reporting problems or significant events.

BER

Basic Encoding Rules - A set of rules for translating ASN.1 values into a stream of octets to be transmitted across a network.

CDP

Control Design Platform.

CDP Controller

Computer (Usually an industrial computer) running CDP application, usually on a true real- time operating system.

CDPUI

The CDP graphical user interface.

Component

Object with strict interface specification.

MIB

Management Information Base. A logical database made up of the configuration, status and statistical information stored at a device. MIB-files are readable text-files.

OID

Object Identifier, a string of numbers derived from a global naming tree, used to identify an object.
E.g. :”1.3.6.1.4.1.16177.1.1.9.1”

SNMP

Simple Network Management Protocol. A protocol using UDP that enables a management station to configure, monitor, and receive trap (alarm) messages from network devices

2. Configuration

Configuration is done by modifying the component xml file inside the Application\Components\ folder. It should not be necessary to modify the model xml file. An example of SNMPManager.xml file (component xml) is found in 4.1.

2.1. NetworkInterface

2.1.1. Description

Encapsulates elements for network, and which UDP ports to use for SNMP.

2.1.2. Example XML

```
<NetworkInterface LocalName="ETH0" SetGetPort="161" TrapPort="162" OwnSrcPort="30543"></NetworkInterface>
```

2.1.3. Elements

Element	Description
SetGetPort	Specifies which destination UDP port which will be used when sending SNMP messages (Set/Get requests) to the agent(s). Default port no for SNMP is 161.
TrapPort	Specifies which UDP port SNMP Manager will listen at to receive Traps from the agent(s). Default port no for SNMP is 162.
OwnSrcPort	Specifies which UDP port SNMP Manager will listen at to receive Get responses from the agent(s). This port will be used as source UDP port when sending SNMP messages (Set/Get requests) to the agent(s). No default value, but must be unique for this application (not used by other sockets).

2.1.4. Required modifications

In 'NetworkInterface' it may not be necessary to modify anything, but OwnSrcPort must be unique for this application.

2.2. Signals

2.2.1. Description

Encapsulates *Signal* elements, where Name is used as a link in GetRequests.

2.2.2. Example XML

```
<Signal Name="Switch1 Temperature" Type="int" Input="0" Unit="deg. C"
  Description="Switch 1 temperature, L400."></Signal>
```

2.2.3. Elements

Ordinary CDP Signals. The Name-elements used here, can be used in the GetRequests-section to get it updated.

2.2.4. Required modifications

Must be updated to wanted signals, and Names must correspond to Signals-items in 'GetRequests'-section.

2.3. Alarms

2.3.1. Description

Encapsulates *Alarm* elements, like Name and level for different alarms.

2.3.2. Example XML

```
<Alarm Name="SNMP Agent Error" Level="Error"
      Enabled="1" Description="Not set yet..."></Alarm>
```

2.3.3. Elements

Ordinary CDP Alarms. The first 4 alarms in the example XML are used for generating alarms when receiving Traps. The next 4 alarms are used for generating alarms if lost contact with the agents. The last alarms have the same Name as some of the auto-generated Alarms from the "ReceiveTraps"-section (see 3.3.1 functionality 2), to be able to give a unique 'Description' for every Alarm.

2.3.4. Required modifications

The first 4 alarms in the example XML must be kept. Update the next alarms according to the agents you have, if you want alarms when loosing contact.

2.4. Parameters

2.4.1. Description

Encapsulates *Parma* elements containing variables which can be edited when installing the application or later from user interface.

2.4.2. Example XML

```
<Parma Name="GenerateStringForUnknowTraps" Unit="" Value="1" DefaultValue="1"
      PreviousValue="0" TimeLastChanged="Tue Oct 18 15:03:07 2005" Description="Variable which decides
      if Alarm-messages shall be generated even if no OID is recognized."></Parma>
```

2.4.3. Elements

Ordinary CDP Parameters. The 2 parameters in the example XML influence on generating alarms when receiving Traps.

2.4.4. Required modifications

The Values for the Parameters may be changed depending on wanted functionality.

2.5. SNMPAgents

2.5.1. Description

Encapsulates *IpMap* elements: a mapping between an agent's AgentNo, IP address, Name, SNMPversion and Alarm (if loosing contact with the agent).

2.5.2. Example XML

```
<IpMap AgentNo="1" IpAddress="10.139.111.5" Name="L400 OnTime Switch 1" SNMPversion="2" Alarm=".L400
Switch1 LostContact"></IpMap>
```

2.5.3. Elements

Value	Description
AgentNo	A unique number which must be used in the SetRequests and GetRequest sections ('AgentNumbers').
IpAddress	IP address of SNMP Agent.
Name	Wanted text/name to be inserted in the Alarm message to identify the sender of the trap.
SNMPversion	Which SNMP version the agent is using. Version 1 and 2 is supported by SNMP Manager.
Alarm	Optional. Which alarm to set when loosing contact with this agent. Must be one of the alarms listed in the "Alarms"-section, specified as "." and the name of the alarm. The agent must be set up to be polled regularly (Interval 1 or 2) if this alarm shall be Set/Cleared. The alarm will be default Set during startup.

2.5.4. Required modifications

The Values must be updated to reflect the agents in the system.

2.6. ReceiveTraps

2.6.1. Description

Encapsulates *Trap* elements, describing which traps to receive, which parameters to decode, what type of Alarm to set, which Alarms to auto-create, which traps shall result in Clear of Alarms, if auto-created Alarms should update corresponding CDP Signals. The Trap-section can basically be configured to work in 3 different ways, see 3 examples below.

2.6.2. Example XML

Functionality 1 (The specified Alarm will be Set, the Alarm's Description will be updated):

```
<Trap TrapOID="1.3.6.1.4.1.705.1.11.11"
      Name="upsmgOnBattery" Alarm=".SNMP Agent Error">
  <Sub OID="1.3.6.1.4.1.705.1.5.1.0" Name="upsmgBatteryRemainingTime"></Sub>
  <Sub OID="1.3.6.1.4.1.705.1.5.2.0" Name="upsmgBatteryLevel"></Sub>
</Trap>
```

Functionality 2 (Auto-creation of unique Alarms, Alarm will be Set, Signal can be updated) :

```
<Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.4"
      Name="Link down" AgentNumbers="1;2;3" Level="Error" Values="1;2;3;4;5;6;7;8"
      SignalOutSet="No routing"></Trap>
```

Functionality 3 (Clear of the auto-created Alarm(s)):

```
<Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.3"
      Name="Link up" LinkedTo="1.3.6.1.4.1.16177.1.2.7.4"></Trap>
```

2.6.3. Elements

See 3.3 for details on how the elements are used:

Functionality 1 (The specified Alarm will be Set, the Alarm's Description will be updated):

Value	Description
TrapOID	An OID to be searched for in the incoming trap-message. The trap will Set specified Alarm.
Name	Text to be exchanged with TrapOID.

Value	Description
Alarm	Which alarm (level) to set. The value should be ".SNMP Agent Error", ".SNMP Agent Warning" or ".SNMP Agent Notify".
Encapsulation of <i>Sub</i> elements	Optional. Contains sub OIDs and corresponding text.

Sub element values (in Trap):

Value	Description
OID	OID to be searched for if TrapOID has been found.
Name	Name/text to be exchanged with OID.

Functionality 2 (Auto-creation of unique Alarms, Alarm will be Set, Signal can be updated) :

Value	Description
TrapOID	An OID to be searched for in the incoming trap-message. The trap will Set an Alarm.
Name	Text to be exchanged with TrapOID. If the auto-created Alarm-name (this Name + AgentNumbers + optional Values) is found in the ordinary Alarm-section, the Alarm-items from the Alarm-section are used, to be able to give e.g unique 'Description' for every Alarm.
AgentNumbers	Which agents to receive the trap from. Auto-creation of Alarms, where AgentNumbers will be used in Alarm-name(s).
Level	Optional. Level of auto-created alarm(s). Default value is Error if not specified.
Values	Optional. Only Integer-values are supported. Values will be used in Alarm-name(s). The integer-value in the received trap will be used to Set correct Alarm.
SignalOutSet	Optional. Auto-creation of Signals, linked to Alarm (Signal-value 0 or 1).

Functionality 3 (Clear of the auto-created Alarm(s)) :

Value	Description
TrapOID	An OID to be searched for in the incoming trap-message. The trap will Clear an Alarm.
Name	Text to be exchanged with TrapOID.
LinkedTo	An OID which must be found in another TrapOID, where the Alarm(s) was auto-created.

2.6.4. Required modifications

The Values must be updated to reflect which traps in the system you want CDP Alarms for.

2.7. SetRequests

2.7.1. Description

Encapsulates *SetReq* elements, describing which OIDs/Names it shall be possible to Set, which variable type it is, which SNMP agents having this OID and which Community to use.

2.7.2. Example XML

```
<SetReq OID="1.3.6.1.4.1.16177.1.2.1.9.0" Name="reset" Type="TYPE_INTEGER" AgentNumbers="1;2;3"
Community="private"></SetReq>
```

2.7.3. Elements

Value	Description
OID	Which OID to use when sending a Set Request to agent(s). The OID must be unique (the same OID value must not be found in other SetReq-lines).
Name	Which name the OID corresponds to. When receiving a SetSNMPvalue (see 3.1) message, the Name in the message will be used when searching the SetReq information to find the corresponding OID. The Name must be unique (the same Name value must not be found in other SetReq-lines).
Type	Specifies what type of variable OID/Name is, see table below for supported types.
AgentNumbers	Which agent(s) this SetRequest can be sent to.
Community	Which community-string to use in this SNMP Set request. 'private' is an often used community-string for SNMP Set requests.

2.7.4. Required modifications

The Values must be updated to reflect which variables at the agent(s) you want shall be possible to Set.

2.7.5. Supported Types in SetReq

Type	Comment
TYPE_BOOLEAN, TYPE_INTEGER	Value in the received message SetSNMPvalue is internally converted to an 'int'.
TYPE_BIT_STRING, TYPE_OCTET_STRING, TYPE_OPAQUE, TYPE_NSAP_ADDRESS, TYPE_OBJECT_IDENTIFIER, TYPE_IP_ADDRESS, TYPE_NULL	Value in the received message SetSNMPvalue is internally converted to an 'char [128]'.
TYPE_COUNTER32, TYPE_GAUGE32, TYPE_TIME_TICKS, TYPE_INTEGER32, TYPE_COUNTER64	Value in the received message SetSNMPvalue is internally converted to an 'uint64_t'.

2.8. GetRequests

2.8.1. Description

Encapsulates *GetReq* elements, describing which OIDs/Names it shall be possible to Get, which variable type it is, which SNMP agents having this OID, which Community to use, which Interval to poll the agent(s) and optional which Signal to update.

2.8.2. Example XML

```
<GetReq OID="1.3.6.1.4.1.16177.1.2.1.1.0" Name="temperature" Type="TYPE_INTEGER"
AgentNumbers="1;2;3" Community="public" Interval="2"
Signal=".Switch1 Temperature;.Switch2 Temperature;.Switch3 Temperature"></GetReq>
```

2.8.3. Elements

Value	Description
OID	Which OID to use when sending a Get Request to agent(s). The OID must be unique (the same OID value must not be found in other GetReq-lines).
Name	Which name the OID corresponds to. When receiving a GetSNMPvalue (see 3.2) message, the Name in the message will be used when searching the GetReq information to find the corresponding OID. The Name must be unique (the same Name value must not be found in other GetReq-lines)

Value	Description
Type	Specifies what type of variable OID/Name is, see table below for supported types.
AgentNumbers	Which agent(s) this GetRequest can be sent to.
Community	Which community-string to use in this SNMP Get request. 'public' is an often used community-string for SNMP Get requests.
Interval	0=The SNMP Manager will only poll agent(s) once (until a response is received), with same frequency as for Interval 1. 1=The SNMP Manager will poll agent(s) with an interval of Interv1 (in PollIntervals). 2=The SNMP Manager will poll agent(s) with an interval of Interv2 (in PollIntervals).
Signal	Optional: Which Signal(s) this GetRequest should update. The value must be ".Name of a Signal specified in the Signal-section". If more than one 'AgentNumbers' are specified for this GetReq, Signal must contain the same number of signals as number of agents. The update-rate of the signal will be equal to specified Interval. If Signal(s) is specified, there are restrictions on supported Types (see below).

2.8.4. Required modifications

The Values must be updated to reflect which variables at the agent(s) you want shall be polled.

2.8.5. Supported Types in GetReq

Type	Comment
TYPE_BOOLEAN, TYPE_INTEGER	Value in the message GetSNMPvalueReply :an internally 'int' is converted to string ('char [128]').
TYPE_BIT_STRING, TYPE_OCTET_STRING, TYPE_OPAQUE, TYPE_NSAP_ADDRESS, TYPE_OBJECT_IDENTIFIER, TYPE_IP_ADDRESS, TYPE_TIME_TICKS, TYPE_NULL	Value in the message GetSNMPvalueReply :a string ('char [128]'). These types are not compatible with CDP Signals.
TYPE_COUNTER32, TYPE_GAUGE32, TYPE_UINTEGER32, TYPE_COUNTER64	Value in the message GetSNMPvalueReply :an internally 'uint64_t' is converted to string ('char [128]').

2.9. PollIntervals

2.9.1. Description

Encapsulates two Interval elements telling the SNMP Manager how often to send the specified Get Requests, also giving update rate for optional Signals.

2.9.2. Example XML

```
<PollIntervals Interv1="10" Interv2="30"></PollIntervals>
```

2.9.3. Elements

Value	Description
Interv1	Number of seconds between every time the SNMP Manager sends Get request of Interval 1.
Interv2	Number of seconds between every time the SNMP Manager sends Get request of Interval 2.

2.9.4. Required modifications

The Values can be changed to get other poll intervals.

2.10. NumIntervForTimeout

2.10.1. Description

Number of interval-periods (since last receiving responses for Get-requests) before pollstatus is set to Lost contact, and the optional Alarm for lost contact is Set. Also the variable's value, and the value of the CDP Signal if specified, is set to 0 when pollstatus is set to Lost contact. In the GetSNMPvalueReply-message, ErrorCode is then set to 3 (see 3.2.2).

2.10.2. Example XML

```
<NumIntervForTimeout>5</NumIntervForTimeout>
```

2.10.3. Elements

No elements, only the value.

2.10.4. Required modifications

May be adjusted to wanted timeout value.

2.11. List of necessary modifications in XML-file

- In 'NetworkInterface' it may not be necessary to modify anything, but OwnSrcPort must be unique for this application.
- The 'Signals'-section must be updated and correspond to Signals-items in 'GetRequests'-section.
- The 'Alarms'-section: If alarms are wanted when loosing contact with agents, update the LostContact alarms (Alarm-item in the 'SNMPAgents'-section). The Alarms equal auto-created alarms can also be updated/removed.
- The Values for the Parameters may be changed depending on wanted functionality.
- The contents in the elements 'SNMPAgents', 'ReceiveTraps', 'SetRequests' and 'GetRequests' should be updated for each unique installation.
- Also the values for 'Interv1', 'Interv2' and 'NumIntervForTimeout' may be changed for wanted functionality.

It can be difficult to know/decide which values to give for *TrapOID* (in Trap element) and *OID* (in Sub element). By using a MIB Browser tool you can investigate the agent's MIB, and find out the OIDs to the traps. By setting the parameter GenerateStringForUnknownTraps to 1, you will get alarm-messages containing the OIDs also, see 3.3.2. Another possibility, is to log traps using an ethernet sniffing tool like Wireshark.

2.12. SNMP Agent Setup

The applications where the SNMP agents are found, must possibly be configured. In some applications, SNMP must first of all be enabled. For most of the SNMP agents, you must specify the IP address(es) of your SNMPManager(s), so the agents are able to send Traps to this/these address(es). In addition, you may specify if the agent shall generate trap or not for specific events.

Example:

L400 OnTime/Westermo ethernet switches: First of all, “IP configuration tool” (a sw tool from supplier) or Web-interface to the switch must be used to enable SNMP. The tool or the Web-interface can be used to enable Port alarm also. To tell the SNMP agent on the switch where to send the traps: Use a MIB Browser tool to set the IP Address of your SNMP Manager, or use your own SNMP Manager to do this: the SNMPManager.xml component file must contain the following line in the SetRequests-section:

```
<SetReq OID="1.3.6.1.4.1.16177.1.2.1.11.0" Name="trapHostAddr1" Type="TYPE_DISPLAYSTRING"  
AgentNumbers="1;2;3" Community="private"></SetReq>
```

(the OID is specific for the switch (found in MIB), the Name is chosen to be similar to what is listed in the MIB, Type is listed in MIB, AgentNumbers: tells which agents (listed as AgentNo in SNMPAgents-section) it shall be possible to send this SetRequest to, Community must correspond with the agent is using). Start your SNMP Manager, use the Web-interface to the SNMP Manager and go the “Messages” section for the SNMPManager component. For the 'SetSNMPvalue' message (see also 3.1), enter e.g. “1;trapHostAddr1;10.0.2.30” (the arguments correspond to AgentNumber;Name;Value) and click on 'Send'. Then the SNMP Manager will send a SNMP Set message to agent number 1 telling that it shall send all traps to the IP address 10.0.2.30 (this should correspond to the IP Address of your SNMP Manager).

3. Functional description

3.1. Set Requests

The SNMP Manager will not send any Set requests to the agent(s) automatically, only when receiving the external message 'SetSNMPvalue'.

3.1.1. SetSNMPvalue-message

The command-type of this message is CM_TEXTCOMMAND. Struct MessageTextCommandWithParmaSend can be used when generating such a message, and struct MessageTextCommandWithParmaReceive is used when receiving/decoding this message. You may also use the Web-interface of your SNMP Manager to send SetSNMPvalue-message, but you don't get any response back. The arguments in the '**SetSNMPvalue**' message are:

"AgentNumber;Name;Value"

E.g. "2;reset;1". Name must be found in one of the SetReq-sections. See 2.7 for details regarding XML-values for SetReq.

3.2. Get Requests

The SNMP Manager will send Get requests to the agent(s) regularly. Which requests and how often are specified in the XML-file, see 2.8 for details. Responses from the agents are received by the SNMP Manager and decoded, and the optional LostContact alarms are Cleared. The received values are either used to update CDP Signals if specified, or just stored internally in the SNMP Manager.

3.2.1. GetSNMPvalue-message

If the external message 'GetSNMPvalue' is received, no SNMP Get request is sent to an agent. The value of the enquired object/variable is fetched from the last value either from the CDP Signal or from the internally stored value. The command-type of 'GetSNMPvalue' is CM_TEXTCOMMAND. Struct MessageTextCommandWithParmaSend can be used when generating such a message, and struct MessageTextCommandWithParmaReceive is used when receiving/decoding this message. You may also use the Web-interface of your SNMP Manager to send GetSNMPvalue-message, but the result in the response is not shown. The arguments in the '**GetSNMPvalue**' message are:

"AgentNumber;Name"

E.g. "3;temperature". Name must be found in one of the GetReq-sections. The SNMP Manager responds with a message called '**GetSNMPvalueReply**'.

3.2.2. GetSNMPvalueReply

The command-type of 'GetSNMPvalueReply' is CM_TEXTCOMMAND. Struct MessageTextCommandWithParmaSend is used when generating this message. The arguments in the 'GetSNMPvalueReply' message is:

“AgentNumber;Name;ErrorCode;ValueType;Value”

ErrorCode:

- 0 = OK,
- 1 = Input parameters and XML parameters not matching,
- 2 = Value has not been polled yet
- 3 = Lost contact with agent (timeout specified by XML element 'NumbIntervForTimeout').
- 4 = Agent returned error or not possible to decode OID value

ValueType:

- 0 = type not supported
- 1 = integer (int) converted to string (char *). Used for TYPE_BOOLEAN, TYPE_INTEGER.
- 2 = string (char *). Used for TYPE_BIT_STRING, TYPE_OCTET_STRING, TYPE_OPAQUE, TYPE_NSAP_ADDRESS, TYPE_OBJECT_IDENTIFIER, TYPE_IP_ADDRESS, TYPE_TIME_TICKS, TYPE_NULL.
- 3 = unsigned integer (uint64_t) converted to string (char *). Used for TYPE_COUNTER32, TYPE_GAUGE32, TYPE_UINTEGER32, TYPE_COUNTER64.

3.3. Traps

When a trap-message is received on the UDP port specified in the component xml file (TrapPort), the message will be decoded.

The component will first search through the received trap and compare the *TrapOIDs* listed in the XML-file with the OIDs in the trap.

If a match is found, the further functionality depends on how the Trap-section is specified in XML (see 3.3.1).

If no match is found, an alarm-message of type “**Unknown SNMP Agent Alarm**” will be generated (see 3.3.2) if the parameter GenerateStringForUnknownTraps is 1. If this parameter is 0, no alarm-message will be generated for unspecified traps.

To enable logging to file or memory of all alarm-messages, you must have the CDPEventManager Add-on.

3.3.1. Functionality for recognized Traps

See 2.6 for description of <Trap>-section description.

Functionality 1 (The specified Alarm will be Set, the Alarm's Description will be updated):

The <Trap>-section must contain an Alarm-element, which refers to an Alarm specified in the <Alarms>-field, like SNMP Agent Error/Warning/Notify. The <Trap>-section must not contain AgentNumbers or LinkedTo. Readable alarm-message is generated by updating the Description of specified Alarm. The Browsers are able to show one (the last received) alarm in each category. When a match between a *TrapOID* and an OID in the received trap-message has been found, the text in the corresponding *Name* element will be exchanged with *TrapOID*. Then the value following the OID will be decoded. Supported types of value are found in 3.3.3.

After this first pair of OID and value is decoded, the component will try to find a match between listed *OIDs* in the optional *Sub* element sections and the OIDs in the received trap. If there are no *Sub* element sections in the *Trap* section, no more decoding is done, even if the received trap contains more pairs of OID and value. If there are *Sub* element sections, the component will search in the received message for match against listed *OIDs*.

For every match, the text in the corresponding *Name* element will be exchanged with *OID*. Then the value following the OID will be decoded. If the received trap-message has other OIDs than listed in the *Sub* element sections, these are not decoded.

An alarm-message of type “SNMP Agent Error/Warning/Notify” contains the following information: “SNMP Trap from:<Name of agent> + IP address, or only IP address, <TrapOID's Name>:<OIDvalue>, <OID's Name>:<OIDvalue>, <OID's Name>:<OIDvalue>.....”.

Example 1:

“SNMP Trap from:UPS 1(10.139.111.126),upsmgOnBattery,upsmgBatteryRemainingTime:12600, upsmgBatteryLevel:100”.

Functionality 2 (Auto-creation of unique Alarms, Alarm will be Set, Signal can be updated) :

The <Trap>-section must contain AgentNumbers-element. The <Trap>-section must not contain Alarms or LinkedTo. Unique Alarms will be auto-created, where Alarm-name will be based on Name, AgentNumbers and optional Values. Remember that max size of Alarm-name is 31. When receiving a trap which is equal to TrapOID, the corresponding alarm will be Set. If the <Trap>-section contains **SignalOutSet="No routing"**, Signals will be auto-created and linked to the Alarm. The Signal-value will be 1 if the Alarm is Set, and 0 if the Alarm is Cleared. The Signal-name will be equal to <Alarm-name>_OutSet (remember max size 31 characters including _OutSet).

Example of Alarm-name:

“Link down Agent1Val1”

If you want to have a unique 'Description' for these Alarms, you must add corresponding Alarms in the ordinary Alarms-section. The Name in the Alarm-section must be the same as the auto-created Name (Name from Trap-section + AgentNumbers + optional Values).

Functionality 3 (Clear of the auto-created Alarm(s)):

The <Trap>-section must contain LinkedTo-element. The <Trap>-section must not contain Alarms or AgentNumbers. The LinkedTo value shall be an OID which must be found in another TrapOID, where the Alarm(s) was auto-created. When receiving a trap which is equal to TrapOID, the corresponding alarm will be Cleared.

3.3.2. Unknown SNMP Agent Alarm

Setting the parameter GenerateStringForUnknownTraps to 1 can be special useful in the starting up period when the contents of the received traps can be unknown/dubiously. Then alarm-messages will be generated even if the *TrapOIDs* listed in the XML-file are not found in the received traps. By investigating the generated alarm-message, the XML-file can be updated with new *TrapOIDs* (and Sub OIDs) if you want this trap to generate an ordinary alarm-message of type “SNMP Agent Error/Warning/Notify”.

An alarm-message of type “Unknown SNMP Agent Alarm” contains the following information:

“SNMP Trap from:<Name of agent> + IP address, or only IP address, <OID>:<OIDvalue>, <OID>:<OIDvalue>, <OID>:<OIDvalue>.....”.

where OID is decoded to the same format as listed in the XML-file, e.g. “1.3.6.1.4.1.16177.1.1.9.1”, and OIDvalue is decoded. Supported types of OIDvalue are found in 3.3.3.

Example (To make it more readable here, a Line Shift has been added after every comma):

SNMP Trap from:R-T200 OnTime Switch 1(10.0.2.86),
 1.3.6.1.2.1.1.3.0:0days+00:43:49.62,
 1.3.6.1.6.3.1.1.4.1.0:1.3.6.1.6.3.1.1.5.4,
 1.3.6.1.2.1.2.2.1.1:4,
 1.3.6.1.2.1.2.2.1.7:1,

1.3.6.1.2.1.2.2.1.8:1,
 1.3.6.1.6.3.1.1.4.3.0:1.3.6.1.4.1.16177.1.1

3.3.3. Supported value types (identifier)

The values following the OIDs in the traps, contain [identifier] [length (of the contents)] [contents]. The identifier tells which type of value is found in the contents field.

Type (identifier)	Corresponding SNMP Byte-value
TYPE_BOOLEAN	0x01
TYPE_INTEGER	0x02
TYPE_COUNTER32	0x41
TYPE_GAUGE32	0x42
TYPE_TIME_TICKS	0x43
TYPE_UINTEGER32	0x47
TYPE_BIT_STRING	0x03
TYPE_OCTET_STRING	0x04
TYPE_OPAQUE	0x44
TYPE_NSAP_ADDRESS	0x45
TYPE_OBJECT_IDENTIFIER	0x06
TYPE_IP_ADDRESS	0x40
TYPE_COUNTER64	0x46

If a value type in the received trap is not found among the supported ones, the value of the parameter ShowOIDvaluesNotSupported decides what to do:

If ShowOIDvaluesNotSupported is 1, the bytes following the OID field will be decoded as following: "OIDvalue-field in HEX:<Byte-value of OID type (identifier)>,<one or more byte(s) giving length of the following OIDvalue>,<OIDvalue (bytes) separated by comma>".

If ShowOIDvaluesNotSupported is 0, the bytes following the OID field is not decoded.

4. Appendix

4.1. Example SNMPManager Component XML-file

```

<?xml version="1.0" encoding="iso-8859-1"?>
<Component Name="SNMPManager" Model="SNMPManager">
  <Debug>3</Debug>
  <Activate>1 </Activate>
  <fs>100</fs>
  <InitialState>Null</InitialState>

  <NetworkInterface LocalName="ETH0" SetGetPort="161" TrapPort="162" OwnSrcPort="30543"></NetworkInterface>

  <Description><![CDATA[A simple SNMP Manager]]></Description>

  <Signals>
    <Signal
      Name="Switch1 Temperature"      Type="int"      Input="0"      Unit="deg. C"
      Description="Switch 1 temperature, L400."></Signal>
    <Signal
      Name="Switch2 Temperature"      Type="int"      Input="0"      Unit="deg. C"
      Description="Switch 2 temperature, L400."></Signal>
    <Signal
      Name="Switch3 Temperature"      Type="int"      Input="0"      Unit="deg. C"
      Description="Switch 3 temperature, L400."></Signal>
    <Signal
      Name="Switch1 PowerSupplyStatus" Type="int"      Input="0"      Unit=""
      Description="Switch 1 power supply status. 1=okpowerAB,2=okpowerA,3=okpowerB"></Signal>
    <Signal
      Name="Switch2 PowerSupplyStatus" Type="int"      Input="0"      Unit=""
      Description="Switch 2 power supply status. 1=okpowerAB,2=okpowerA,3=okpowerB"></Signal>
    <Signal
      Name="Switch3 PowerSupplyStatus" Type="int"      Input="0"      Unit=""
      Description="Switch 3 power supply status. 1=okpowerAB,2=okpowerA,3=okpowerB"></Signal>
    <Signal
      Name="UPS Battery Remaining Time" Type="int"      Input="0"      Unit=""
      Description="The time remaining actual charge vs actual load (dynamic)."></Signal>
    <Signal
      Name="UPS Battery Level"         Type="int"      Input="0"      Unit=""
      Description="The battery level as a percentage of charge."></Signal>
    <Signal
      Name="UPS Battery Voltage"       Type="int"      Input="0"      Unit=""
      Description="The actual battery voltage."></Signal>
    <Signal
      Name="UPS Battery Current"       Type="int"      Input="0"      Unit=""
      Description="The actual battery current."></Signal>
    <Signal
      Name="UPS Battery Temperature"   Type="int"      Input="0"      Unit=""
      Description="The battery temperature."></Signal>
  </Signals>

  <Alarms>
    <Alarm Name="SNMP Agent Error"      Level="Error"
      Enabled="1" Description="Not set yet..."></Alarm>
    <Alarm Name="SNMP Agent Warning"    Level="Warning"
      Enabled="1" Description="Not set yet..."></Alarm>
    <Alarm Name="SNMP Agent Notify"     Level="Notify"
      Enabled="1" Description="Not set yet..."></Alarm>
    <Alarm Name="Unknown SNMP Agent Alarm" Level="Notify"
      Enabled="1" Description="Not set yet..."></Alarm>
    <Alarm Name="L400 Switch1 LostContact" Level="Error"
      Enabled="1" Description="Set when lost contact with L400 Switch1."></Alarm>
    <Alarm Name="L400 Switch2 LostContact" Level="Error"
      Enabled="1" Description="Set when lost contact with L400 Switch2."></Alarm>
    <Alarm Name="L400 Switch3 LostContact" Level="Error"
      Enabled="1" Description="Set when lost contact with L400 Switch3."></Alarm>
    <Alarm Name="UPS1 LostContact"      Level="Error"
      Enabled="1" Description="Set when lost contact with UPS 1."></Alarm>
    <Alarm Name="Link down Agent1 Val1" Level="Error" Enabled="1"
      Description="Link down for port 1 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val2" Level="Error" Enabled="1"
      Description="Link down for port 2 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val3" Level="Error" Enabled="1"
      Description="Link down for port 3 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val4" Level="Error" Enabled="1"
      Description="Link down for port 4 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val5" Level="Error" Enabled="1"
      Description="Link down for port 5 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val6" Level="Error" Enabled="1"
      Description="Link down for port 6 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val7" Level="Error" Enabled="1"
      Description="Link down for port 7 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent1 Val8" Level="Error" Enabled="1"
      Description="Link down for port 8 on Switch 1." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent2 Val1" Level="Error" Enabled="1"
      Description="Link down for port 1 on Switch 2." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent2 Val2" Level="Error" Enabled="1"
      Description="Link down for port 2 on Switch 2." SignalOutSet="No routing"></Alarm>
    <Alarm Name="Link down Agent2 Val3" Level="Error" Enabled="1"
      Description="Link down for port 3 on Switch 2." SignalOutSet="No routing"></Alarm>
  </Alarms>

```

```

    Description="Link down for port 3 on Switch 2." SignalOutSet="No routing"></Alarm>
    Name="Link down Agent2 Val4" Level="Error" Enabled="1"
    Description="Link down for port 4 on Switch 2." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent2 Val5" Level="Error" Enabled="1"
    Description="Link down for port 5 on Switch 2." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent2 Val6" Level="Error" Enabled="1"
    Description="Link down for port 6 on Switch 2." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent2 Val7" Level="Error" Enabled="1"
    Description="Link down for port 7 on Switch 2." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent2 Val8" Level="Error" Enabled="1"
    Description="Link down for port 8 on Switch 2." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val1" Level="Error" Enabled="1"
    Description="Link down for port 1 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val2" Level="Error" Enabled="1"
    Description="Link down for port 2 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val3" Level="Error" Enabled="1"
    Description="Link down for port 3 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val4" Level="Error" Enabled="1"
    Description="Link down for port 4 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val5" Level="Error" Enabled="1"
    Description="Link down for port 5 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val6" Level="Error" Enabled="1"
    Description="Link down for port 6 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val7" Level="Error" Enabled="1"
    Description="Link down for port 7 on Switch 3." SignalOutSet="No routing"></Alarm>
  <Alarm Name="Link down Agent3 Val8" Level="Error" Enabled="1"
    Description="Link down for port 8 on Switch 3." SignalOutSet="No routing"></Alarm>
</Alarms>

<Parameters>
  <Parma Name="GenerateStringForUnknownTraps" Unit="" Value="1" DefaultValue="1"
    PreviousValue="0" TimeLastChanged="Tue Oct 18 15:03:07 2005"
    Description="Variable which decides if Alarm-messages shall be generated even if no OID is
    recognized."></Parma>
  <Parma Name="ShowOIDvaluesNotSupported" Unit="" Value="1" DefaultValue="1"
    PreviousValue="0" TimeLastChanged="Tue Oct 18 15:59:47 2005"
    Description="Variable which decides if not supported OID values shall be decoded or not."
    ></Parma>
</Parameters>

<SNMPAgents>
  <IpMap AgentNo="1" IPAddress="10.0.2.89" Name="L400 OnTime Switch 1" SNMPversion="2"
    Alarm=".L400 Switch1 LostContact"></IpMap>
  <IpMap AgentNo="2" IPAddress="10.0.2.84" Name="L400 OnTime Switch 2" SNMPversion="2"
    Alarm=".L400 Switch2 LostContact"></IpMap>
  <IpMap AgentNo="3" IPAddress="10.0.2.86" Name="L400 OnTime Switch 3" SNMPversion="2"
    Alarm=".L400 Switch3 LostContact"></IpMap>
  <IpMap AgentNo="4" IPAddress="10.0.2.87" Name="UPS 1" SNMPversion="1"
    Alarm=".UPS1 LostContact"></IpMap>
</SNMPAgents>

<!--
  All enterprise TrapID must be defined before TrapID from standard MIBs.
  If not, enterprise Trap OIDs must be defined as Sub OIDs under TrapOID="1.3.6.1.6.3.1.1.4.1.0"
-->
<ReceiveTraps>
  <Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.1"
    Name="Status warning" AgentNumbers="1;2;3" Level="Warning"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.2"
    Name="Status no warning" LinkedTo="1.3.6.1.4.1.16177.1.2.7.1"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.4"
    Name="Link down" AgentNumbers="1;2;3" Level="Error" Values="1;2;3;4;5;6;7;8"
    SignalOutSet="No routing"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.16177.1.2.7.3"
    Name="Link up" LinkedTo="1.3.6.1.4.1.16177.1.2.7.4"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.705.1.11.17"
    Name="UPSutPowFailed" AgentNumbers="4" Level="Error" SignalOutSet="No routing"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.705.1.11.18"
    Name="UPSutPowRestored" LinkedTo="1.3.6.1.4.1.705.1.11.17"></Trap>
  <Trap TrapOID="1.3.6.1.4.1.705.1.11.11"
    Name="upsmgOnBattery" Alarm="SNMP Agent Error">
    <Sub OID="1.3.6.1.4.1.705.1.5.1.0" Name="upsmgBatteryRemainingTime"></Sub>
    <Sub OID="1.3.6.1.4.1.705.1.5.2.0" Name="upsmgBatteryLevel"></Sub>
  </Trap>
  <Trap TrapOID="1.3.6.1.6.3.1.1.4.1.0"
    Name="SNMP Trap" Alarm="SNMP Agent Error">
    <Sub OID="1.3.6.1.6.3.1.1.5.1" Name="Coldstart"></Sub>
    <Sub OID="1.3.6.1.6.3.1.1.5.2" Name="Warmstart"></Sub>
    <Sub OID="1.3.6.1.6.3.1.1.5.3" Name="Link Down"></Sub>
    <Sub OID="1.3.6.1.6.3.1.1.5.4" Name="Link Up"></Sub>
    <Sub OID="1.3.6.1.2.1.2.2.1.1" Name="Port"></Sub>
    <Sub OID="1.3.6.1.2.1.2.2.1.7" Name="AdminStatus"></Sub>
    <Sub OID="1.3.6.1.2.1.2.2.1.8" Name="OperStatus"></Sub>
  </Trap>
</ReceiveTraps>

<SetRequests>
  <SetReq OID="1.3.6.1.4.1.16177.1.2.1.9.0" Name="reset" Type="TYPE_INTEGER"
    AgentNumbers="1;2;3" Community="private"></SetReq>
  <SetReq OID="1.3.6.1.4.1.16177.1.2.1.11.0" Name="trapHostAddr1" Type="TYPE_DISPLAYSTRING"
    AgentNumbers="1;2;3" Community="private"></SetReq>
  <SetReq OID="1.3.6.1.4.1.16177.1.2.1.12.0" Name="trapHostAddr2" Type="TYPE_DISPLAYSTRING"
    AgentNumbers="1;2;3" Community="private"></SetReq>
  <SetReq OID="1.3.6.1.4.1.705.1.4.19.0" Name="upsmgConfigAlarmAudible_y1_n2" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public"></SetReq>
</SetRequests>

```

```

<!-- Number of seconds between every time the SNMP Manager sends Get request of specified Interval.
      If CDP Signals are specified, they will be updated with this interval. -->
<PollIntervals Interv1="10" Interv2="30">/PollIntervals>

<!-- Number of interval-periods (since last receiving responses for Get-requests) before pollstatus is
set to Lost contact. -->
<NumbIntervForTimeout>5</NumbIntervForTimeout>

<GetRequests>
  <GetReq OID="1.3.6.1.4.1.16177.1.2.1.1.0" Name="temperature" Type="TYPE_INTEGER"
    AgentNumbers="1;2;3" Community="public" Interval="2"
    Signal=".Switch1 Temperature;.Switch2 Temperature;.Switch3 Temperature">/GetReq>
  <GetReq OID="1.3.6.1.4.1.16177.1.2.1.11.0" Name="trapHostAddr1" Type="TYPE_DISPLAYSTRING"
    AgentNumbers="1;2;3" Community="public" Interval="0">/GetReq>
  <GetReq OID="1.3.6.1.4.1.16177.1.2.1.12.0" Name="trapHostAddr2" Type="TYPE_DISPLAYSTRING"
    AgentNumbers="1;2;3" Community="public" Interval="0">/GetReq>
  <GetReq OID="1.3.6.1.4.1.16177.1.2.1.10.0" Name="powerSupply" Type="TYPE_INTEGER"
    AgentNumbers="1;2;3" Community="public" Interval="1"
    Signal=".Switch1 PowerSupplyStatus;.Switch2 PowerSupplyStatus;.Switch3
    PowerSupplyStatus">/GetReq>
  <GetReq OID="1.3.6.1.4.1.705.1.5.1.0" Name="upsmgBatteryRemainingTime" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public" Interval="2"
    Signal=".UPS Battery Remaining Time">/GetReq>
  <GetReq OID="1.3.6.1.4.1.705.1.5.2.0" Name="upsmgBatteryLevel" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public" Interval="2" Signal=".UPS Battery Level">/GetReq>
  <GetReq OID="1.3.6.1.4.1.705.1.5.5.0" Name="upsmgBatteryVoltage" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public" Interval="2" Signal=".UPS Battery Voltage">/GetReq>
  <GetReq OID="1.3.6.1.4.1.705.1.5.6.0" Name="upsmgBatteryCurrent" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public" Interval="2" Signal=".UPS Battery Current">/GetReq>
  <GetReq OID="1.3.6.1.4.1.705.1.5.7.0" Name="upsmgBatteryTemperature" Type="TYPE_INTEGER"
    AgentNumbers="4" Community="public" Interval="2"
    Signal=".UPS Battery Temperature">/GetReq>
</GetRequests>
</Component>

```

4.2. SNMP Protocol

For information only, the basics of the SNMP Protocol is listed here:

4.2.1. Introduction

An SNMP based network management system incorporates the following:

- SNMP Manager(s)/Management station(s)
- SNMP Agents
- MIB
- Network management protocol (SNMP over UDP)

There are 5 main types of messages that are exchanged between a manager and an agent:

get-request : manager fetches the value of a variable from agent

get-next-request : fetch the value of the next variable

set-request : manager sets the value of a variable

get-response : agent returns the value of a variable to manager

Trap : agent notifies the manager when something happens, e.g. an attached interface has gone down

The 3 first types are using destination UDP port 161 (default), the get-response is using UDP port 161 (default) as source port (the destination port is the same as the source port used in the request), while the Trap messages are sent with destination port 162 (default).

4.2.2. Format

The principle of BER is that each field has an introducer that indicates the datatype of the contents and its length. The basic pattern used to encode a value is :

```
[identifier] [length (of the contents)] [contents]
```

The identifier declares the datatype of the contents. This datatype can be simple (such as an INTEGER) and then it is defined as a primitive type, or it can be more complex (such as a SEQUENCE). In this last case, each content field is introduced by its own identifier and length :

```
[identifier SEQUENCE] [length SEQUENCE]
  [identifier field 1] [length field 1] [value field 1]
  [identifier field 2] [length field 2] [value field 2]
  ...
  [identifier field N] [length field N] [value field N]
[SEQUENCE] (optional)
```

SNMP messages are constructed using the SEQUENCE datatype. Each MIB variable that is carried in a SNMP message is required to be a simple datatype.

Simple description of SNMP Protocol format (all v1+v2 messages, except v1 trap):

Version	Community	PDU type	Ident	Error status	Error index	OID 1	Value 1	OID n	Value n
---------	-----------	----------	-------	--------------	-------------	-------	---------	-------	-------	---------

SNMP version1 protocol format for Trap PDU type:

Version	Community	PDU type	Enter-prise	Agent Addr	Gen Trap	Spec Trap	Time Stamp	OID 1	Value 1	OID n	Value n
---------	-----------	----------	-------------	------------	----------	-----------	------------	-------	---------	-------	-------	---------

The value-field following the OIDs, contain [identifier] [length (of the contents)] [contents]. The identifier tells which type of value is found in the contents field.