SNMP

SNMP protocol

Supported device types and versions
Communication line configuration
Communication station configuration
I/O tags configuration
Messages of Trap type receiving and processing
Browsing and reading the tree of values from the script
Literature
Changes and modifications
Document revisions

Supported device types and versions

SNMP protocol (Simple Network Management Protocol) is used for monitoring and administration of network components. It allows detection of the network devices' status and changing of their settings. In an application, it is possible to monitor the functionality of e.g. routers, switches, computers, etc.

To create a station equipped with SNMP protocol, it is necessary to have a UDP line (line of TCP/IP-UDP type). It is worth mentioning here that a TCP/IP-UDP line in the perception of the D2000 system is actually a UDP socket which is a logical device to support communication of individual stations. It is not possible to use stations with a different protocol on a line where stations with SNMP protocol exist!

Communication line configuration

- Communication line category: TCP/IP-UDP.
- UDP parameters:
  - Host: There are three ways:
    1. The IP address of the particular network interface – datagrams will be transmitted and received only via this interface.
       Example: *192.168.1.10*
    2. The symbolic name of a particular network interface.
       Example: *D2SRV_PRIMARY*
    3. *ALL* -the configured UDP port is opened on all available network interfaces. An optimal network interface should be used for communication based on routing tables. The reception of messages will be performed on all network interfaces.
  - Port: UDP port number (0 through 65535) from which the D2000 KOM process sends requests and receives the responses. If the value is 0, the port number is assigned automatically by OS.
    Note: Ports 161 and 162 are the standard UDP ports used in SNMP but they are often reserved for SNMP agents - that is why it is recommended to choose different ports. Problems can occur with value 0 (zero) if the network uses firewalls and other security measures. Then a specific port needs to be configured on firewalls so that the packets from this port are passed via firewalls.

Note:
If SNMP protocol needs to run in a redundant system, where two instances of the D2000 KOM process are running concurrently on two different computers and the IP address cannot be positively determined in the line configuration, it is appropriate to choose „ALL“ configuration option or to name the network addresses identically as e.g. SNMP_LAN and assign them a correct IP address in the *hosts* file of each computer. See example:

on Computer 1:	192.168.0.1	SNMP_LAN
on Computer 2:	192.168.0.11	SNMP_LAN

Protocol parameters on the line

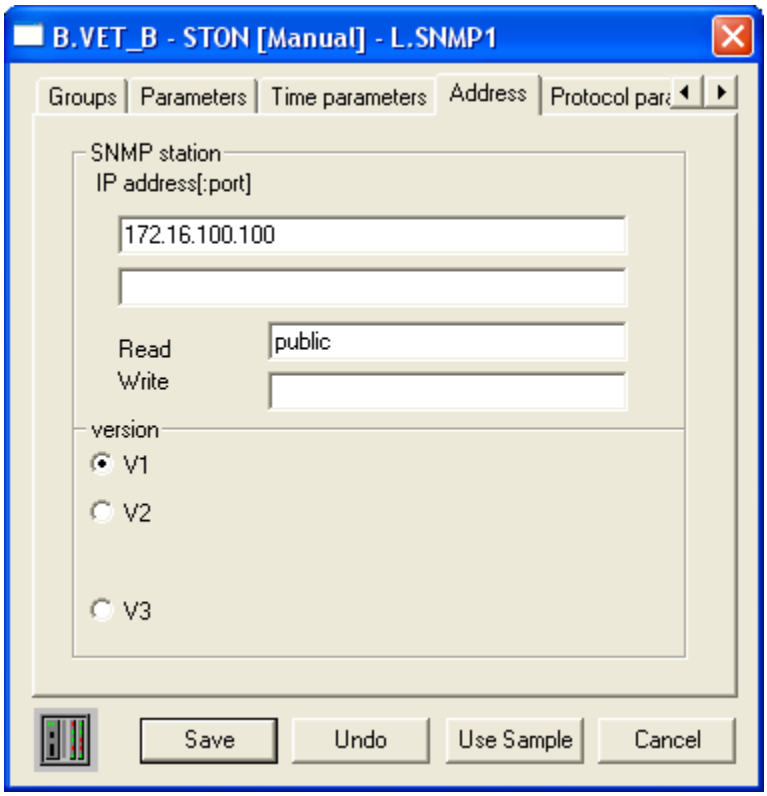
The following parameters of the protocol can be set on the line:

Keyword	Full name	Description	Unit	Default value

<table>
<tr><td>TRACE</td><td>Trace Level</td><td><table>
<tr><td>Trace level = 0</td><td>- no debugging information output, the same as turning it off in Line parameters</td></tr>
<tr><td>Trace level = 1</td><td>- only information on receiving and sending UDP packet and IP address</td></tr>
<tr><td>Trace level = 2</td><td>- adds information on request preparation</td></tr>
<tr><td>Trace level = 3</td><td>- adds packet's HEX dump</td></tr>
<tr><td>Trace level = 4</td><td>- the same as the value of 3</td></tr>
<tr><td>Trace level = 5</td><td>- adds:
detailed analysis of packet structure in ASN1 coding
order of data in the packet
detailed information</td></tr>
<tr><td>Trace level = 9999</td><td>- adds information on preparation and decision making of packet distribution and that concerning searching</td></tr>
</table>
The values 5 and 9999 are intended for debugging and their permanent use is not recommended. In case, that the information is needed from a monitored station(s) only, the setting of the Trace level can be performed for a particular station in its configuration dialog box.

The value of 1 is recommended for ordinary operation.</td><td>-</td><td>1</td></tr>
<tr><td>TE</td><td>Trap Enable</td><td>Enables to receive the messages of the Trap type.</td><td>Boolean</td><td>False</td></tr>
<tr><td>TTI</td><td>Trap IP Address</td><td>The IP address for receiving the Trap messages.</td><td>-</td><td>ANY</td></tr>
<tr><td>TTP</td><td>Trap Port</td><td>UDP port for receiving the Trap messages.</td><td>-</td><td>162</td></tr>
</table>

Communication station configuration



- Communication protocol: **SNMP Manager**.
- Station's address: it is defined in format IP_address1[:port1], IP_address2[:port2].

IP_address may be set in decimal dotted notation (e.g. 192.168.0.1) or as a name (e.g. *SrvMoxa1*), which assumes address translation by means of DNS or a *hosts* file. Address1 and Address2 concern the existence of primary and backup lines/routes. Address 2 is usable for example for a server containing two network interface cards, which is connected to two different network segments available via two different network paths.

Port is a number in range 1..65535 on which an SNMP agent expects communication to take place. As the default (if not stated, or set to 0) port the standard port 161 will be used.

Note:

- If the line has only a primary IP address configured (numerical or symbolic), UDP packets are sent from this address to both IP addresses of the station. One numerical primary IP address of line + two IP addresses of the station are valid for network topology where the local network is non-redundant but the remote network (where the station is located) is accessible via two redundant communication paths.
- If the line has both IP addresses configured, UDP packets to IP_address1 leave from the primary IP address of the line, and UDP packets to IP_address2 leave from the backup IP address of the line.
  The situation when e.g. IP_address1 is not configured conforms to the topology when the station is connected to a backup communication path only.

Protocol
The employed version of SNMP protocol – one of the options can be selected:

- SNMP_V1 – the oldest version – does not support any secured access to the SNMP agent. It only distinguishes the objects that are freely accessible (public) and those belonging to a restricted group (private).
- SNMP_V2 – a version that supports authentication to access individual data types - an agent might (not) provide a particular set of data for an anonymous user (a manager,...) and different data for a user whose identity has been verified by entering a correct name and password.
- SNMP_V2C – the same as SNMP_V2 – the D2000 system does not distinguish these variants.
- SNMP_V3 – so far the latest protocol version – besides functions provided by SNMP_V2C, supports functions for authentication and encryption. It requires entering the name of an authentication server and authentication keys, to authenticate prior to communication with an agent, and keys for encrypting communication.

SNMP_V2, SNMP_V2C, and SNMP_V3 are not supported yet. Neither the writing into SNMP agent nor reading MIB branches as a table (structured I/O tags or directly entered structure entries) are supported.

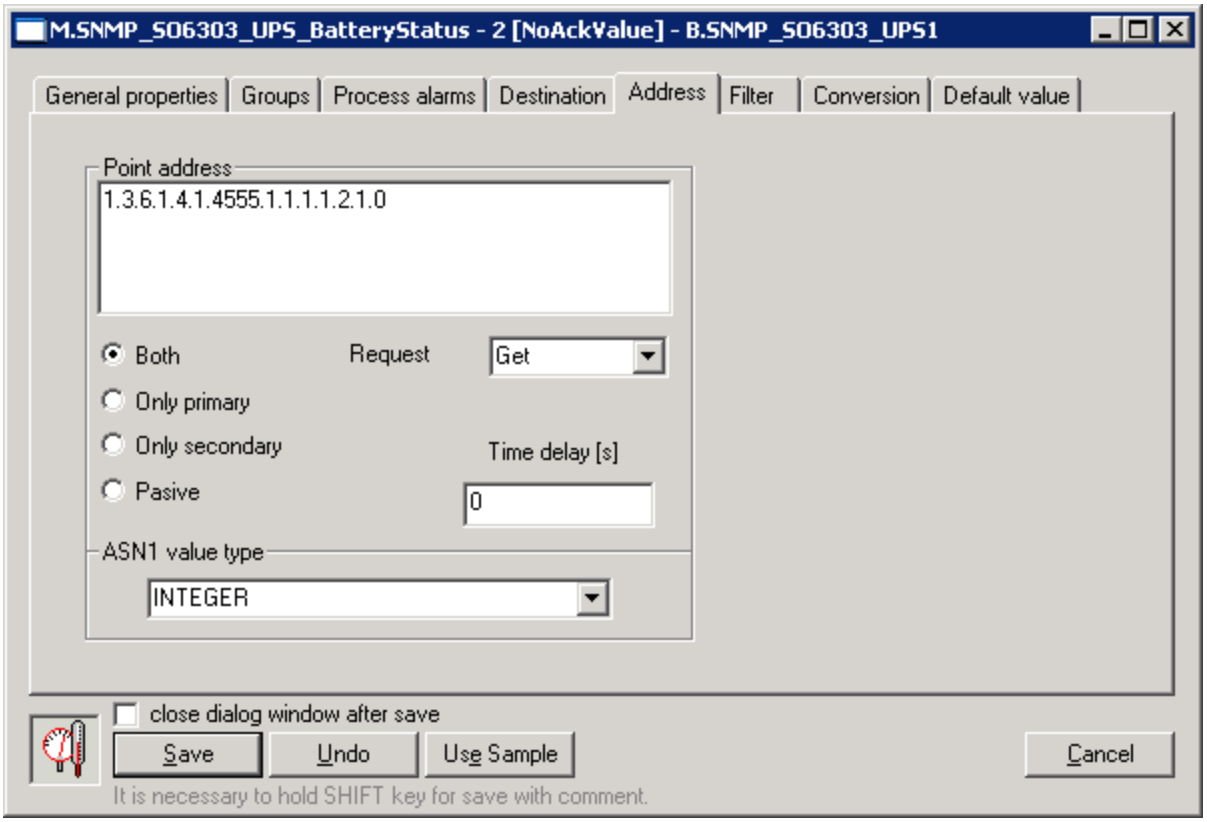
Station protocol parameters

The following station protocol parameters can be set:

Table no. 2

Keyword	Full name	Description	Unit	Default value
WT	Wait Timeout	The timeout period for the response to the read request.	ms	
RC	Retry Count	A number of re-sent read requests, before the reading is considered to be unsuccessful and another I/O tag will be queried.	-	
EC	Max Error Count	Maximum count of unsuccessful read requests, until the station changes its value to StCOMERR state. A successful value delivery nullifies all counters and puts the station back into StON state.	-	
TL	Trace Level	The same meaning as parameter Trace Level on a line, but this setting is valid for the particular station. However, the higher value of a line parameter Trace level takes precedence. Note: Debugging of incoming packets is influenced by the line parameter Trace Level because at the time of reception it is still unknown which station the packet belongs to.	-	

I/O tags configuration



Address1: Address of I/O tag. The address specifies the OID {Object identifier) of an object. It is displayed in a numerical format, e.g.: 1.3.6.1.2.1.1.1.0.

An I/O tag with such an address will all be read via a network path, which is currently operational (a primary or a backup line is determined according to the result of a reply to a previous request or possibly can be switched automatically).

If it would be necessary to have information on whether the primary or backup IP address of the device is available, it is possible to use the so-called forced addressing by selecting the option *Only Primary*, resp. *Only backup*. This will ensure that the I/O tag value will be acquired only from the primary, resp. backup station address. The *Both* option is the standard option, where the values of the I/O tags are obtained continuously from both addresses of the station (if they are configured). The *Passive* option means that the value of the I/O tag is not read directly, but is obtained indirectly as a copy of the value of another I/O tag with the same address, but in the active mode e.g. *Only primary*.

If the object with a specified OID address does not exist, the SNMP agent returns an error code with a different OID address (because the object with the required OID does not exist) and therefore the communication will be denoted as unsuccessful. The I/O tag passes to the „Unknown value“ state. If it is necessary to indicate the line status by value change and not by the validity of the object's value, the object of DI type can be created, an integer value (e. g. UpTime) can be asked for and an automatic number to boolean conversion can be utilized, where 0 is converted to false and the other values to True. The I/O tag can be then configured to use a default value and to set the default to False. Then the object may acquire only the values True or False, depending on the object's availability in the SNMP agent.

The I/O tag with an address starting with *%IGNORE* will be ignored.

Request: Default value *Get* causes the values to be read by a *Get* SNMP request.
Some devices have problems providing values by the *Get* request if the object is an item of an array. Then, you must configure the type of request GetNext and the address should be OID of the previous object (to find the address, use the Java application MIB Browser (http://tl1.ireasoning.com/mibbrowser.shtml) that reads the whole tree of values and detects the OID address of the previous object).

Time delay: Offers a possibility to set a delay period for particular I/O tags – to optimize the network's load. This time is added to the current time after a successful reading of the I/O tag's value and the next request will be processed as soon as the current time is greater than or equal to the time calculated in this way.
If the object's value is unknown, the object will be included in communication in the next periodic request (according to the time parameters of the station) regardless of the delay time.
The time delay parameter does not influence the processing of TRAP messages if the TRAP has the same address as the /O tag.

After receiving the value from the SNMP agent, the conversion will be done according to the real type of value in the SNMP protocol and the required type in the D2000 system. If it is not possible to carry out the conversion, the value will be invalid and a report about the wrong conversion will be logged into a trace file.

ASN1 value type: Specifies, the value type in the SNMP agent's response. It also determines applicable conversions. The value type can be detected in the MIB database (*note:* MIB database browser is not a part of the solution). One of the freely available browsers can be used and desired data format can be set based on the obtained information. It is recommended to use the MIB Browser Java application (http://tl1.ireasoning.com/mibbrowser.shtml).

Possible value types:

Integer	input value - expected as a signed integer number (up to 64bit *****)
Unsigned	input value - expected as an unsigned integer number (up to 64bit *****)
Float	input value - expected as a floating-point number (float, a long float)
Text	input value - text string
IP address	the input sequence of bytes interpreted as a sequence of numbers separated by a dot – the sequence is converted to text
Hex text	the input sequence of bytes is interpreted as a sequence of hexadecimal numbers separated by a colon – the sequence is converted to text

The value types *IP address* and *Hex text* can be applied to an arbitrary input data type, which will be further handled as a sequence of bytes. E.g., the input value of text type with value "test@ipesoft.sk" can be interpreted in the following ways:

Text: "test@ipesoft.sk"
IP address: „112.101.114.105.99.104.64.105.112.101.115.111.102.116.46.115.107“7“
Hex text: „70:65:72:69:63:56:40:69:70:65:73:6F:66:74:2E:73:6B“

These methods were introduced to support cooperation with IP and MAC addresses of network interfaces.

***** D2000 system supports values of objects in the maximum range of 32 bits for signed integer types. Therefore, if the number is bigger, then the maximum value of the 32-bit range will be assigned to it. If the input object of the D2000 system is of *Ai* type, the system will attempt to convert it to *Real*.

Permissible types: **Di**, **Ci**, **Ai**, **TxI**, **TiR**, **TiA**

The following table shows the supported conversions of value types:

	Typ hodnoty v systéme D2000					
Typ hodnoty SNMP	*Di*	*Ci*	*Ai*	*TxI*	*TiR*	*TiA*
Boolean	●	●	●	●		
Integer	●	●	●	●		
Unsigned	●	●	●	●		
Counter	●	●	●	●		
Gauge	●	●	●	●		
Float		●	●	●		
Text				●		
TimeTicks		●	●	●	●	
Time			●	●		●

- supported conversion

Trap messages receiving and processing

SNMP protocol also allows, except for cyclic value reading, to send messages about important events. These messages are called Traps.
SNMP agent sends the Traps to the configured IP address and port (by default 162) which is configured (simple devices support sending of Traps to one IP address and port, advanced ones send Traps to more addresses).
The Trap IP address parameter must be configured to activate a task that receives the Traps on the Trap port.
Trap receiving is supported in the version V1 and V2C of protocol SNMP. By default, one device sends Traps using one version of the protocol.
To receive Traps from a particular device, I/O tags with the following text addresses must be configured on the station (however, there is no need to configure all of them):

Text addresses of I/O tags for Traps in SNMP protocol, version V1:

I/O tag address	Data type	Description
TRAP_ENTERPRISE	OID	OID of the object which generate Trap (for particular device it is constant). **Note:** A producer of device can be often detected from OID.
TRAP_GENERIC_TRAP	Integer	Identifier of Trap class. Following values are defined in RFC 1157 for SNMP, version 1: • 0 - coldStart • 1 - warmStart • 2 - linkDown

		• 3 - linkUp • 4 - authenticationFailure • 5 - egpNeighborLoss • 6 - enterpriseSpecific
TRAP_SPECIFIC_TRAP	Integer	Specific code of message.
TRAP_TIMESTAMP	TimeTicks	Time-stamp (according to RFC 1157 it means the hundreds of second that passed between the last network reinitialization of device and trap generating. **Note:** If I/O tag is *Ai - Analog input*, its value will be in seconds, i.e. TimeTicks/100. If I/O tag is *Ci - Integer input*, its value will be in hundreds of second, i.e. TimeTicks. The maximum value for integer value in D2000 is 2^31-1 (because the integer type is implemented as 32-bit integer with sign). I/O tag of *Ci - Integer input* type cannot acquire the higher values than 2^31-1. According to RFC 1157, the Time-stamp is of TimeTicks type which is a non-negative integer. It can acquire higher values than 2^31-1 which are not allowed to be written into I/O tag of *Ci - Integer input* type. That is why it is recommended to configure I/O tag of *Ai - Analog input* type.
TRAP_OID	OID	OID of object that caused a formation of Trap or object which Trap relate to.
TRAP_VALUE	Arbitrary	Value of object that caused a formation of Trap or object which Trap relate to. **Note 1:** Because the value is arbitrary, it is recommended to configure I/O tag of *TxtI - Text input* type. Otherwise, some values will not be converted (e.g. to *Integer input*) and value TRAP_VALUE will not be changed. **Note 2:** Trap can contain several couples (OID, value) as well. In this case, the value of I/O tags with addresses TRAP_OID and TRAP_VALUE will be set for all couples step-by-step. It is possible to configure event which is initiated when the value of I/O tag with address TRAP_VALUE is changed, and to save the couples (OID, value) into database.
TRAP_CONFIRM	Boolean	I/O tag which confirm the values processing. Because several couples (TRAP_OID, TRAP_VALUE) can exist in one Trap message, the correct processing by e.g. ESL script needs so that KOM process will set next couple after the first one is processed. Also the values of other input I/O tags for Trap messages should be set after signalization that previous values have been already processed. If the output I/O tag with address *TRAP_CONFIRM* exists, KOM process will set next couple of input I/O tag values after it is written into output I/O tag with address *TRAP_CONFIRM* (ESL script will execute the record as one of the last operations). The values of another I/O tags (with addresses TRAP_ENTERPRISE, TRAP_GENERIC_TRAP, TRAP_SPECIFIC_TRAP, TRAP_TIMESTAMP and TRAP_OID) will be set if it is the processing of the first couple of values (TRAP_OID, TRAP_VALUE). In case of another couples, the values of I/O tags will be the same and they will be changed during the next Trap message processing. If the output I/O tag with address *TRAP_CONFIRM* does not exist, the values of all input I/O tags with addresses TRAP_* will be set immediately after Trap message occurred. The values can get lost, because of existence of the several value couples in Trap message or because of new message arrival, before the user script has processed the previous values.

Text addresses of I/O tags for Traps in SNMP protocol, version V2C:

I/O tag address	Data type	Description
TRAP_REQUEST_ID	Integer	Increment number of Trap.
TRAP_ERROR_STATUS	Integer	Error code. Default value is zero (0) but it can acquire one of the following values (see RFC 1448): • noError(0) • tooBig(1) • noSuchName(2) • badValue(3) • readOnly(4) • genErr(5) • noAccess(6) • wrongType(7) • wrongLength(8) • wrongEncoding(9) • wrongValue(10) • noCreation(11) • inconsistentValue(12) • resourceUnavailable(13) • commitFailed(14) • undoFailed(15) • authorizationError(16) • notWritable(17) • inconsistentName(18)
TRAP_ERROR_INDEX	Integer	Extended error code (often it is 0).

TRAP_UP TIME_OID	OID	OID of object SysUpTime.0. This item should have the value 1.3.6.1.2.1.1.3.0 according to RFC 1448. But, if the item has not get this value in the implementation, the value can be find out by I/O tag with the address *TRAP_UPTIME_OID*.
TRAP_UP TIME_VAL UE	TimeTi cks	Value of object sysUpTime. The Note, mentioned in description of address TRAP_TIMESTAMP, is valid for this value.
TRAP_TR AP_OID	OID	OID of object SnmpTrap.0. This item should have the value 1.3.6.1.6.3.1.1.4.1.0 according to RFC 1448 (i.e. OID of object snmpTrapOID, see RFC 1450). But, if the item has not get this value in the implementation, the value can be find out by I/O tag with the address *TRAP_TRAP_OID*.
TRAP_TR AP_OID_V ALUE	OID	Identifier of Trap category, meaning of which corresponds to item TRAP_GENERIC_TRAP in SNMP, version V1, but it is of OID type that allows to define the error codes, specific for particular producers and devices. Meaning of standard OID, which can acquire according to RFC 1450, are following: • 1.3.6.1.6.3.1.1.5.1 - coldStart • 1.3.6.1.6.3.1.1.5.2 - warmStart • 1.3.6.1.6.3.1.1.5.3 - linkDown • 1.3.6.1.6.3.1.1.5.4 - linkUp • 1.3.6.1.6.3.1.1.5.5 - authenticationFailure • 1.3.6.1.6.3.1.1.5.6 - egpNeighborLoss • 1.3.6.1.6.3.1.1.5.7 - enterpriseSpecific
TRAP_OID	OID	The same meaning as TRAP_OID in SNMP, version V1.
TRAP_VA LUE	arbitrary	The same meaning as TRAP_VALUE in SNMP, version V1.
TRAP_CO NFIRM	Boolean	The same meaning as TRAP_CONFIRM in SNMP, version V1.

Note 1: It will be sufficient to configure the input I/O tags with addresses TRAP_OID, TRAP_VALUE and output I/O tag with address TRAP_CONFIRM to confirm the value processing.

Note 2: If the parameter *Trap enable* has been already configured on the line, the individual task will be activated because of Trap messages processing. This task will receive the messages on the chosen UDP port, number of which specifies a link parameter *Trap port* (default 162).

If the Trap message processing is configured on the line with address *ANY* or *ALL* and on the particular port, it is not possible to configure the Trap message processing on another line and use the same port. It causes a collision. But it is possible to configure another parameter *Trap port* (e.g. 163) and set, on the devices, the sending of this messages to another port (e.g. 163).

Note 3: In a redundant system, user must take into consideration that SNMP agents usually support the sending traps to just one IP address (set in advance). Therefore, when redundancy is applied, everything will be ready for receiving traps on the side of D2000 system, but the monitored devices will send traps to the original address. A support of DDNS could be a solution but only in case that SNMP agent can use DNS services.

User must ensure so that the lines will not use the same network interface on the same UDP port. A line with IP address configuration as ANY basically causes blocking (restricting) UDP port on all network interfaces, which may collide with another TCP-UDP line.

Browsing and reading the tree of values from the script

The version D2000 7.02.006 and higher supports the dynamic address change of I/O tag by TELL command SETPTADDR. This address together with I/O tag address GETNEXT_OID allow to browse and read the whole tree of values by SNMP request GetNext.

I/O tag address	Value type	Description
GETNEXT _OID	TxtI - Text input	OID of next object, it is in the response on request GetNext. Only requests that have been generated as the result of address change of I/O tag by tell command SETPTADDR are taken into consideration and not the requests that have been generated as a result of cyclic reading of I/O tags.

To read the tree of values, you should configure two input I/O tags of *TxtI - Text input* type. One of them has the special address *GETNEXT_OID.* Tell command SETPTADDR set the address of the second I/O tag.
After the address is set the KOM process will generate the request to read the I/O tag. If the request *GetNext* is in address (e.g. *SETPTADDR M. MySnmpVariable 1.3.6.1.2.1.1 TYPE=3;RQ=1)*, the OID (sent with reply) will be recorded into I/O tag with address *GETNEXT_OID* (e.g. 1.3.6.1.2.1.1.1.0). After that, the new tell command containing this address (*SETPTADDR M.MySnmpVariable 1.3.6.1.2.1.1.1.0 TYPE=3;RQ=1*) can be sent and so on.

Example of ESL script that shows the browsing and reading the first 100 objects from tree starting with address 1.3.6.1.2.1.1 and recording the OID addresses and values into the structure *_objlist*:

```
ENTRY query_device_OnClick
  INT _ret
  TIME _t
  TEXT _currOID  ; OID of object prior to object being read
  INT _obj_count ; number of read objects
```

```
  RECORD (SD.OID_Value) _objlist ; structure for storing OID+value of read objects

  _obj_count := 0
  _currOID := "1.3.6.1.2.1.1" ; start browsing the tree from successor of this OID

  DO_LOOP
    _t := M.SNMP_VariableAddress\TIM ; remember original time
    _ret := COMMAND "SETPTADDR M.SNMP_VariableAddress " + _currOID + " TYPE=3;RQ=1" ON SELF.KOM
    EXIT_LOOP _ret # _ERR_NO_ERROR

    DO_LOOP ; wait till the time of variable changes
      EXIT_LOOP _t # M.SNMP_VariableAddress\TIM
      DELAY 1[ms]
    END_LOOP

    EXIT_LOOP ! M.SNMP_VariableAddress\VLD ; invalid - error reading value from SNMP

    _obj_count := _obj_count + 1
    REDIM _objlist[_obj_count]
    _objlist[_obj_count]^OID := M.SNMP_GetNextOid ; OID of object
    _objlist[_obj_count]^Value := M.SNMP_VariableAddress ; value of object

    EXIT_LOOP _obj_count > 100    ; I need only first 100 values
    _currOID := M.SNMP_GetNextOid ; OID of the object which came with GetNext request
  END_LOOP
END query_device_OnClick
```

Literature

RFC
http://www.ietf.org/rfc.html
http://www.rfc-editor.org/rfcsearch.html

SNMP
http://www.snmplink.org
http://www.simpleweb.org/ietf/rfcs/rfcbymodule.html
http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp?topic=/com.ibm.ztpf.doc_put.01/gtpc1/gtpc1m0a.htm
http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=32
http://www.microsoft.com/technet/archive/winntas/maintain/featusability/networkm.mspx?mfr=true

ASN.1
http://asn1.elibel.tm.fr/en/introduction/index.htm
http://asn1.elibel.tm.fr/en/standards

Changes and modifications

Document revisions

- 20. 3. 2006 - 1. version (testing version)
- 31. 7. 2007 - 2. version (SNMP in asynchronous mode)
- 16. 1. 2009 - GetNext support

Related pages:

Communication protocols