SNMP Quick Start and Introduction

October 2015
An ICONICS Whitepaper
www.iconics.com

Visualize Your Enterprise™

CONTENTS

1 ABOUT THIS DOCUMENT _____ 1

1.1 SCOPE OF THE DOCUMENT 1
1.2 COPYRIGHT AND CONFIDENTIALITY 1
1.3 REVISION HISTORY 1

2 SNMP PROTOCOL _____ 2

2.1 INTRODUCTION 2
2.2 SNMP BASIC COMPONENTS 2
2.3 SNMP BASIC COMMANDS 3
2.4 MANAGEMENT INFORMATION BASE (MIB) 3
2.5 INSTALL SNMP ON THE MACHINE 5
2.6 SNMP TRAPS MANAGEMENT 5

3 SNMP CONNECTOR _____ 5

3.1 INTRODUCTION 6
3.2 SNMP CONFIGURATOR INTERFACE 7
3.3 CREATING THE DATABASE 8
3.4 PARSING MIB FILES 10
3.5 EXPLORE THE NETWORK 14
3.6 ORGANIZE THE COLLECTED DATA 23
3.7 CREATE YOUR DISPLAY IN GRAPHWORX32 27

4 SNMP SIMULATOR _____ 32

4.1 SNMP AGENTS BASICS 32
4.2 ICONICS SNMP SIMULATOR FEATURES 32
4.3 USING ICONICS SNMP SIMULATOR 32

5 CONCLUSIONS _____ 35

5.1 WHAT SNMP CONNECTOR CAN OFFER 36

1 About This Document

1.1 Scope of the Document

There is no comprehensive document available to users that address the configuration issues and detailed method to set up the SNMP environment for GENESIS32. This document is written to introduce ICONICS Representatives, Distributors, System Integrators and Customers with the upcoming technology and provide user training. This document not only gives some basic information about the SNMP Protocol, but helps in creating a display in GraphWorX32 that shows useful SNMP information coming from the network.

1.2 Copyright and Confidentiality

1.3 Revision History

V9 - May 2006

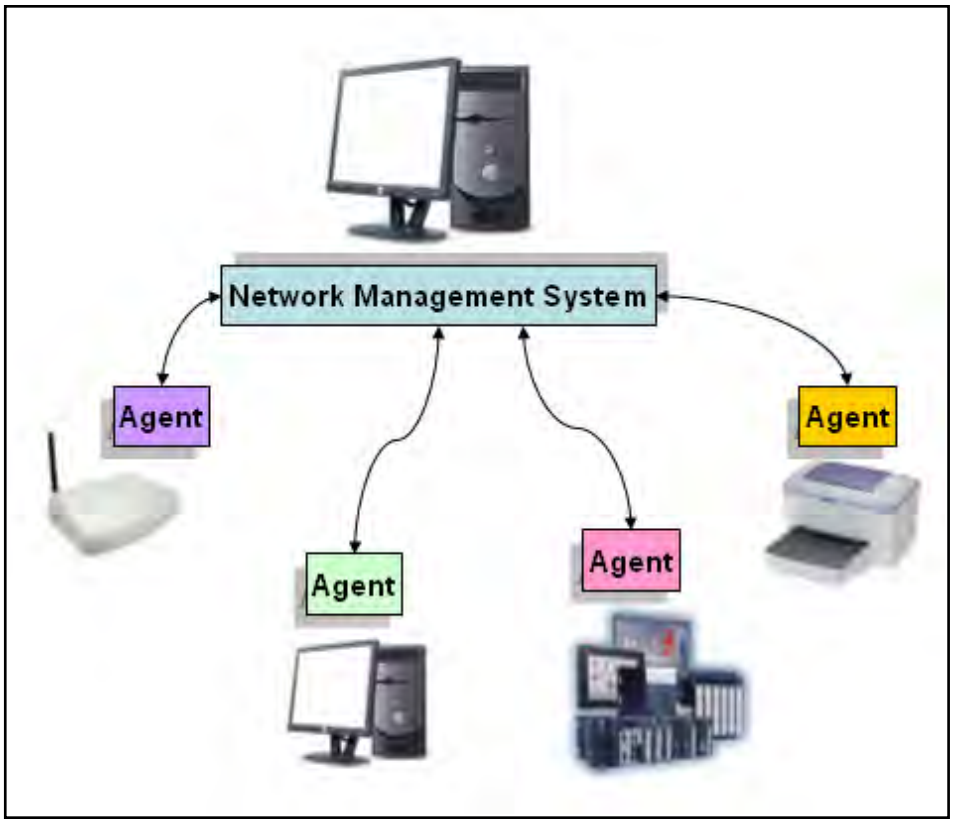
2 SNMP Protocol

2.1 Introduction

SNMP stands for Simple Network Management Protocol, and is a simple protocol that allows devices to expose useful information to other devices. This information can be the CPU fan speed of a computer or the routing table of the router. Almost every network device answers to SNMP requests. SNMP gives Network Managers access to information from nearly every device connected to the Network.

2.2 SNMP Basic Components

An SNMP-managed network consists of three key components: **managed devices, agents, and network-management systems (NMSs)**. A managed device is a network node that contains an SNMP agent and that resides on a managed network. Managed devices collect and store management information and make this information available to NMSs (like the GENESIS32 SNMP Connector) using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.



Network Management System

An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP. An NMS executes applications

that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.

2.3 SNMP Basic Commands

Managed devices are monitored and controlled using four basic SNMP commands: **read**, **write**, **trap**, and **traversal operations.**

The **read** command is used by an NMS to monitor managed devices. The NMS examines different variables that are maintained by managed devices.

The **write** command is used by an NMS to control managed devices. The NMS changes the values of variables stored within managed devices.

The **trap** command is used by managed devices to asynchronously report events to the NMS. When certain types of events occur, a managed device sends a trap to the NMS.

Traversal operations are used by the NMS to determine which variables a managed device supports and to sequentially gather information in variable tables, such as a routing table.

2.4 Management Information Base (MIB)

The Manager and the Agent exchange useful data, but how are these data organized?

To simplify everything, let's use the File System Management as an example. In the file system the data are structured in folders and files: "C:\windows\system32\ping.exe" is the location of the file "ping.exe" which is inside the folder "system32", which is inside the folder "windows", which is placed in the "C:" partition.

In SNMP the structure of the information is really similar to this one. For example, "*iso.org.dod.internet.mgmt.mib-2.system.sysDescr*" is the location for "*sysDescr"* (the description of the system), which is inside the folder "*system*", which is inside "*mib-2*", etc.

Unfortunately it is impossible to store names into the device because it would be harder to manage them automatically for the Manager. So instead of the names, we found numbers inside the device. For example, "*iso.org.dod.internet.mgmt.mib-2.system.sysdescr*" becomes "*1.3.6.1.2.1.1.1*". This notation is called OID (Object Identifier) and identifies the information "*Description of the system*".

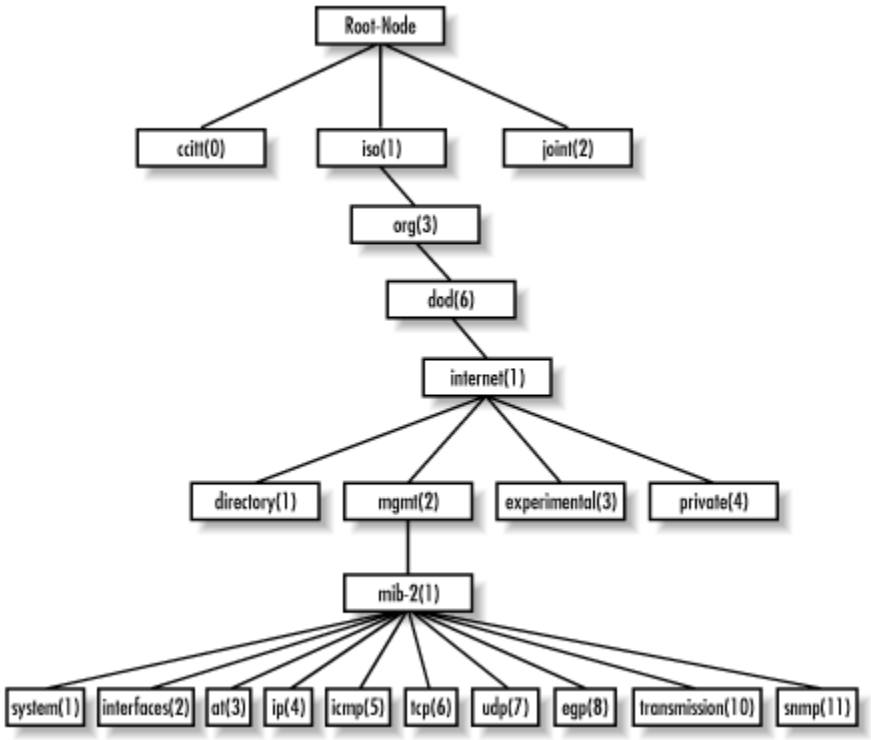
The set of OIDs stored into a device is called MIB (Management Information Base), which represents all the data that the device can expose through SNMP. In the MIB of a printer, for example, there will be the toner level, the text which is displayed on the screen, the number of pages printed, etc. Each piece of information will have a specific OID.

Obviously it is impossible for a human to understand what "*1.3.6.1.2.1.1.1*" means, and that is why there are text files that say how to convert these numbers into their respective names. Since the set of OIDs is called MIB, these files are called MIB files.

They not only translate numbers into names, but they also give more useful information about the OIDs. For example, in the MIB file that converts "*1.3.6.1.2.1.1.1*" to "*iso.org.dod.internet.mgmt.mib-2.system.sysDescr*" there is also written that sysDescr is a string, that it is read only, and that it is "*A textual description of the entity. This value should include the full name and version identification of the system"s hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.*"

There is a set of OIDs that is standard. It must be the same on all the devices and it is created by ISO organization. ISO also created the MIB files related to these OIDs. Each vendor can implement its own MIB on a device. These vendors also deploy the MIB files to allow users to read this non-standard information from the device.

Two types of managed objects exist: scalar and tabular. **Scalar objects** define a single object instance. **Tabular objects** define multiple related object instances that are grouped in MIB tables. SysDescr, for example, is a scalar object.



Root of the Standard MIB Tree

2.5 Install SNMP on the Machine

There are some basic steps to set up a starting SNMP configuration. Feel free to modify it to improve your network security.

- Install GENESIS32 Version 9.
- If it is not already installed, please install SNMP from the Windows CD. To do this select on **Add or Remove programs** in the Control Panel. Then click on **Add\Remove windows components.** From here install **Management Tools.**
- Open the **Services** list and look for **SNMP Service** and **SNMP Trap Service.**
- Double-click on the first one to configure it. Verify if the service is started; if not, start it.
- Select the **Traps** tab and set “public” as the community string; then add your IP address to the list of trap destinations. You can even add other IP addresses if you wish to send traps also to other computers in your network.
- Now select the **Security** tab and check **Send authentication trap** check box. In the upper box you have to add two communities: **public,** with read-only permission, and **private,** with read-write permission. In the other box select **Accept SNMP packets from any hosts.** Click **OK** to close the window.
- Double-click on **SNMP Trap Service.** Verify that the service is running; if not, click **Start.**
- *Verify that you do not have any other trap-receiving software running, because this would create some conflicts with ICONICS software.*

2.6 SNMP Traps Management

Traps are alarms automatically sent through SNMP messages by a device. These messages can say that the "*Printer door is open*" as well as "*The machine has been reset*". Thanks to the translation of the SNMP connector there is no practical difference between traps and OPC alarms management within the GENESIS32 suite.

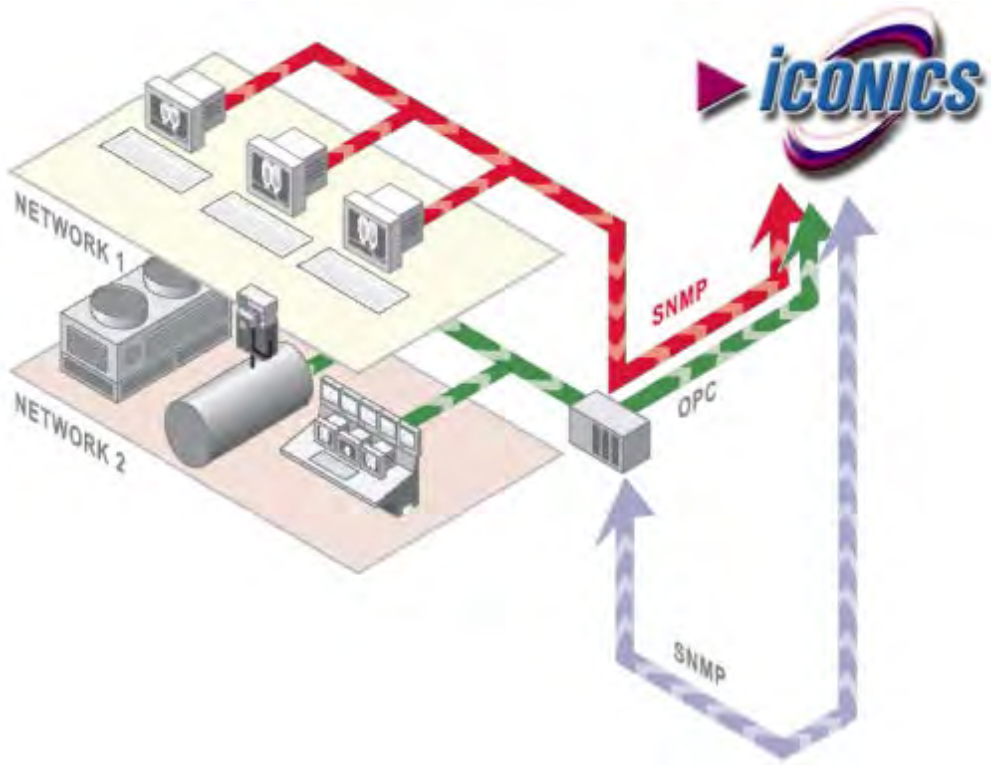
To receive traps, it is necessary to follow three simple steps:

- **Configure your machine:** As we have seen in the previous section, it is necessary to set up the SNMP Windows service to receive traps. This means that we must insert an IP address for every device that we want to manage to receive traps from that device. So if we want to receive all the traps from the printer whose IP Address is *10.1.2.100,* we must insert the IP Address *10.1.2.100* in the **Traps** tab of the **Windows SNMP Service** dialog, as explained in the previous section.
- **Configure the device:** Set up the device to let it know where to send the traps. Follow the device instructions to set the IP address of the machine as the destination of the traps.
- **Configure SNMP Connector:** The **SNMP Configurator** is a filter from the devices and the GENESIS32 suite. For each device it is possible to enable or disable the traps reception. This may seem a repetition of the previous steps, but simplifies the traps enabling/disabling. (It is not necessary to change every time the configuration on the devices and on the Windows SNMP Service). SNMP Connector

3 SNMP Connector

3.1 Introduction

SNMP Connector is a set of applications that allows you to create your own Network Management Systems through GENESIS32. This allows you to manage the Network (through SNMP) and perform Supervisory Control and Data Acquistion (SCADA through OPC) with the very same application. Moreover, you can monitor and manage the Ethernet-based PLC-control networks through SNMP for a complete and universal network management.



Network Management Using SNMP Connector

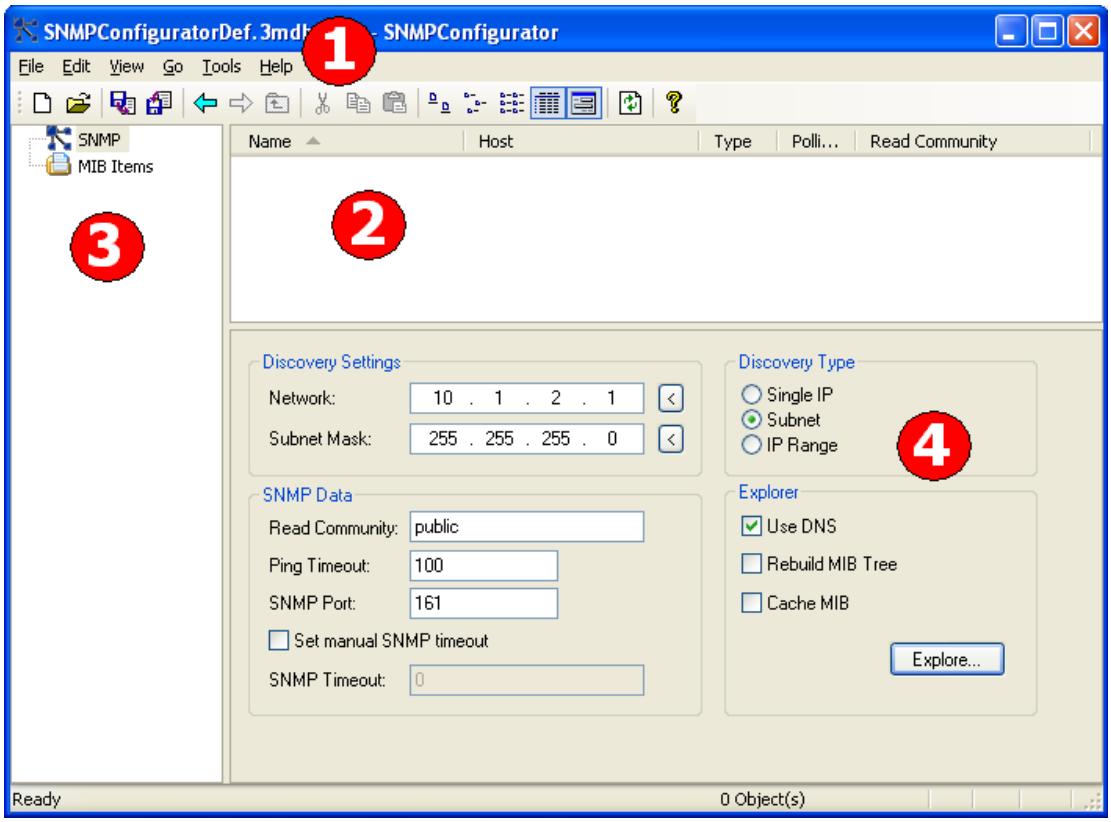
The SNMP Connector Suite consists of:

- **SNMP MIB Parser:** It is possible to parse the MIB files containing all the information about the OIDs and the devices.
- **SNMP Explorer:** Through a fast network scan, SNMP Explorer retrieves all the SNMP agents over a network and shows you all the information contained in the MIB through a simple tree structure.

- **SNMP Configurator:** This is the core of the SNMP Connector. Here you can organize all the information retrieved through SNMP and modify the SNMP tags.

3.2 SNMP Configurator Interface

To open the SNMP Configurator, select on **Start -> ICONICS Tools -> SNMP Configurator.**



Empty SNMP Configurator

There are four main areas in the SNMP Configurator:

1. **Menu and Toolbar:** The buttons of the menus and the toolbar allow you to create, open or save a database, to modify items and to manage the SNMP Configurator interface (showing or hiding the dialog, for example).
2. **Upper right panel:** When a folder is selected, all the objects contained in the folder will be shown here together with the main information about the objects.
3. **Left Panel:** There are two empty trees in the left panel. The **SNMP** tree will contain all the SNMP tags (we will see later what SNMP tags are), while the **MIB Items** tree is a "dictionary": it will contain the translations between the

numbers and the names (and all the information about the names). Unlike traditional dictionaries, the structure of this one is a tree, because of the tree structure of the SNMP data. Thanks to this it will be simpler to find an item inside this MIB "dictionary".

4. **Lower right panel:** From the dialogs of SNMP Configurator you can to visualize and modify the data about the SNMP objects, as well as launch the SNMP Explorer or parse new MIB files.

3.3 Creating the database

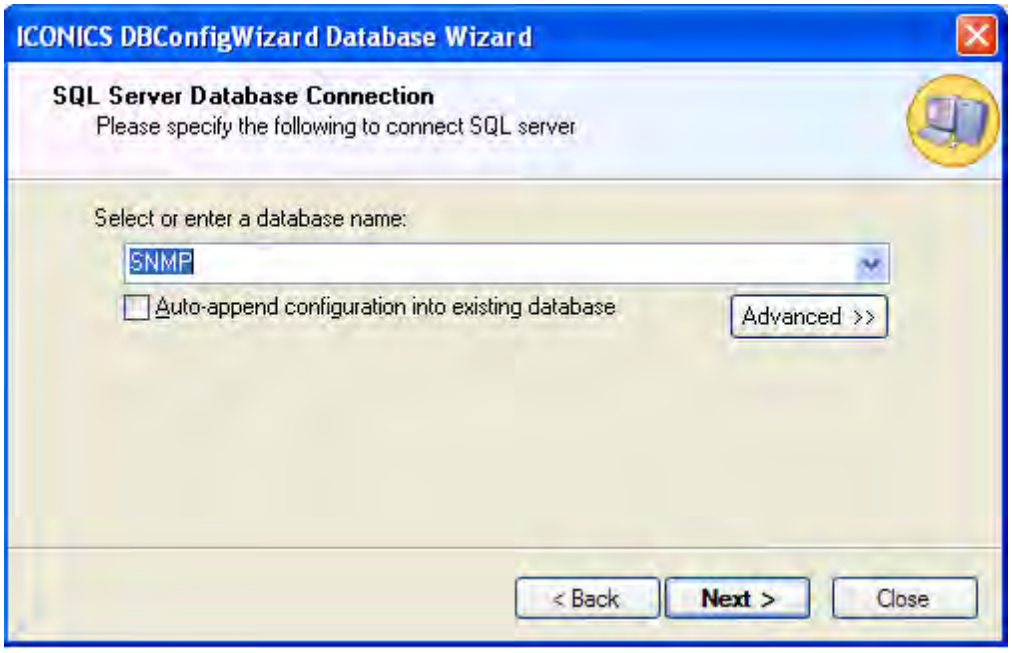
Click on **File -> New** to create a new database. This will launch the SNMP Configurator Database Wizard.

SNMP Configurator Database Wizard

You can to store the SNMP data in a Microsoft SQL database or a Microsoft Access database.

Selecting a Database Type

If you choose SQL database, you must set the right connection to the SQL database.



Specifying a SQL Database Name

Otherwise you can to choose the right path for an Access database.

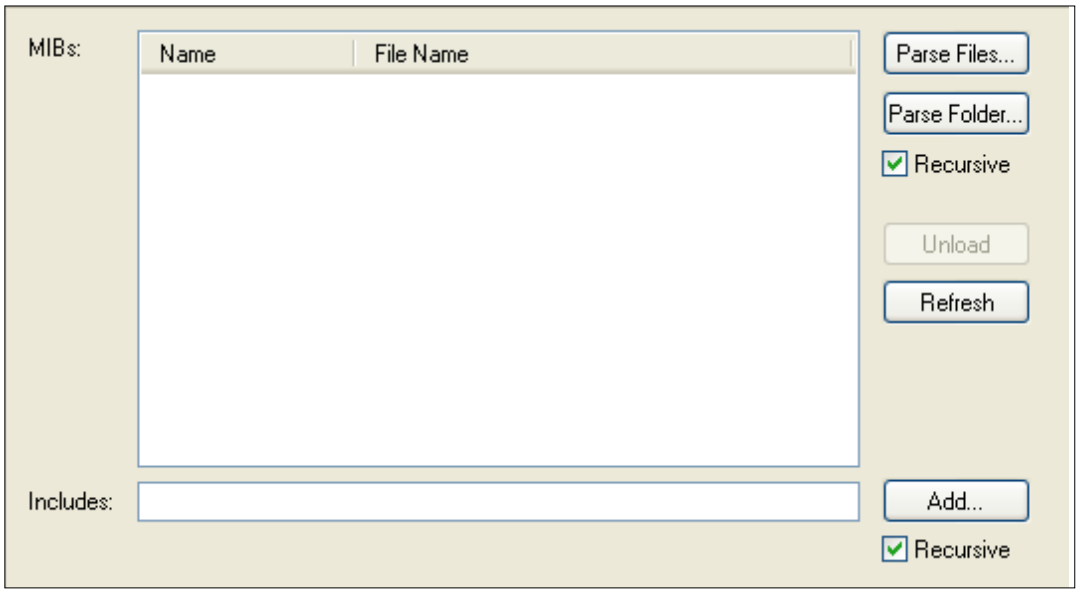
Specifying an Access Database Name

3.4 Parsing MIB Files

Click on **Tools -> MIB Manager** or on the MIB Items Tree.

Launching the MIB Manager

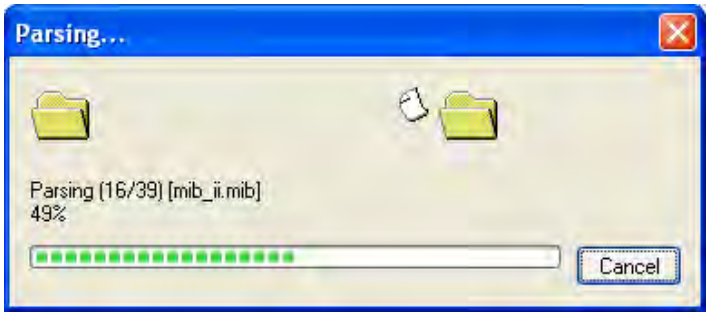
The MIB Manager Dialog will appear.



MIB Manager

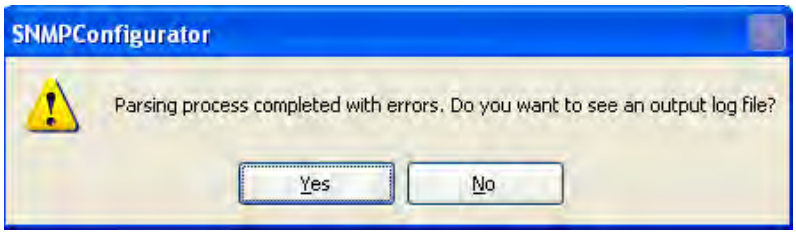
You can parse a single MIB file (**Parse Files** button), or a whole folder (**Parse Folder**) with its subfolders (check the **Recursive** check box). You can also add an **Include** folder (eventually with its subfolders) where the parser will search for useful MIB files that will help the parsing process to be completed. Once the folder is selected the parser automatically collects information from the selected MIB files and puts everything into the database.

During the parsing process a dialog appears and gives a feedback about what is happening.



Parsing in Progress

If there are some errors in the MIB files, you can see a log.

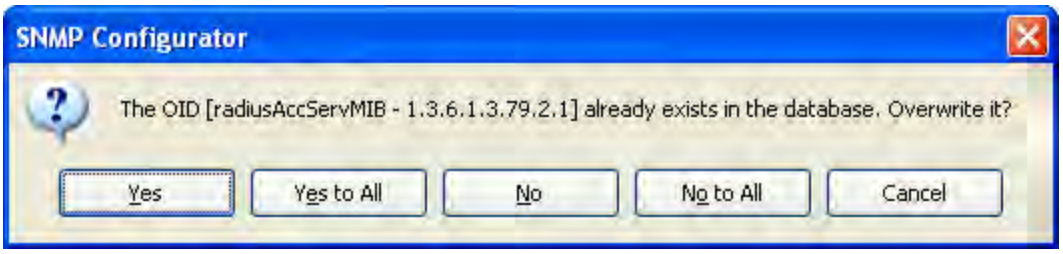


Option to See Log File

You can also store in the database the information collected.

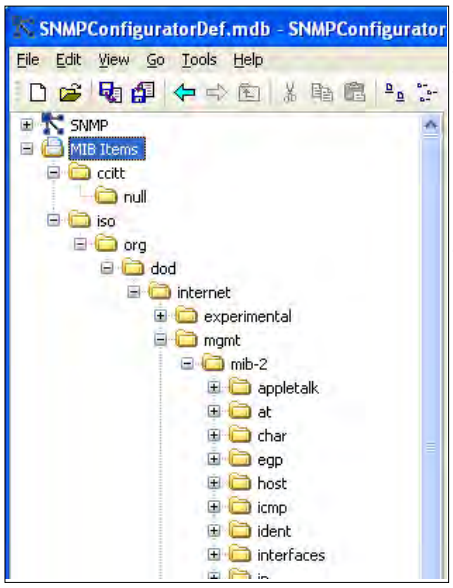
Storing Resolved MIB Items to Database

Sometimes OIDs are defined many times in different files. If an OID is already stored in the database and a MIB file redefines it, the parser asks you which copy of the OID must be stored in the database.



Overwriting OIDs to the Database

At the end of the process, in the Parser dialog you can see the list of the MIB files successfully parsed and whose OIDs are stored in the database. The files are not linked to the **MIB Items** tree. The list is needed only to show you the files already parsed.



When the Parsing process is completed all the information can be accessed by the **MIB Items** tree, where they are organized into a hierarchical tree structure.

The MIB tree gives a translation between the OIDs names (which are human readable strings) and the OIDs address (which are a sequence of numbers and dots).

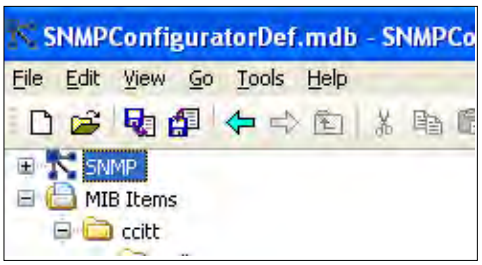
Click on every object in the tree view to see all the information about the object in the List View and the Dialog View. You can also modify such information.

We'll see how this translation is really useful in the next section: "Explore the Network".

To populate the MIB tree you can start parsing the "C:\windows\system32\" folder, which contains some standard MIB files and the Microsoft MIB files.

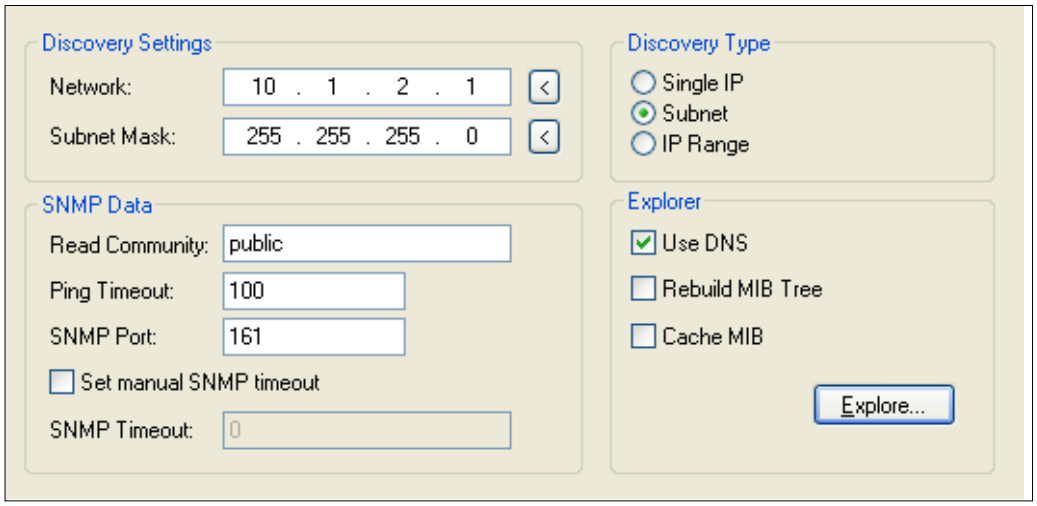
3.5 Explore the Network

Click on the **SNMP** tree control.



SNMP Tree Control

In the Dialog View you will see the SNMP Discovery dialog.



SNMP Discovery Dialog

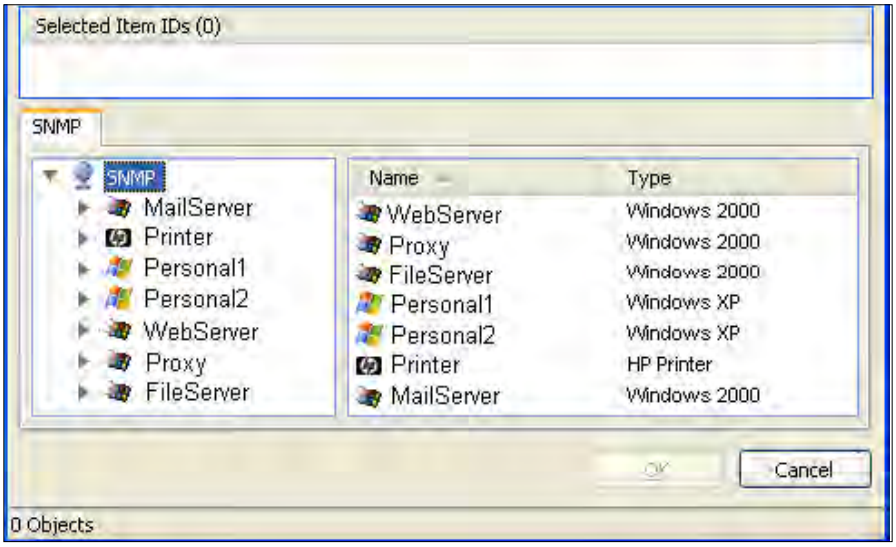
From here you can change the main settings to optimize the Network discovery.

- Changing the **Discovery Type** you can access a single device through SNMP, to search all the SNMP-enabled devices within a subnet or between two IP addresses (for example, from 192.168.1.1 to 192.168.10.255). The Subnet or the IP Addresses must be set into the **Discovery Settings.**
- Clicking on the arrows near the IP Address fields, you can automatically fill the fields with pre-defined data (like all the subnet masks available or the IP addresses of the local machine).
- The **Read Community** is a kind of password that allows you to access all the agents where the very same community is set. In other words, the

devices that have a different Read Community from the one set in the dialog will not be shown after the Network Discovery.

- **Ping Timeout:** The discovery function will discard all the devices that answer after this timeout.
- **SNMP Port:** The discovery tries to connect through this port to retrieve SNMP data.
- **SNMP Timeout:** You can set a different timeout for the SNMP requests (for example, because the SNMP agent speed is lower than the device speed to the ping requests).
- **Use DNS:** Use this check box to show the names of the devices instead of their IP Address.
- **Rebuild MIB Tree:** When the data are collected through SNMP Explorer, all the information will be stored in the SNMP tree. You can organize the data like a standard MIB tree by checking this check box. Otherwise you will organize the data by creating your own folders.
- **Cache MIB:** You can retrieve the whole tree instead of exploring it step by step. This is more time consuming but more reliable. In fact, sometimes the devices do not implement correctly the MIB tree, and to visualize correctly all the data it is necessary to retrieve the whole MIB of the device locally and then reconstruct the MIB tree automatically.

Click **Explore...** to start the discovery of the network. A new dialog appears with all the devices found.



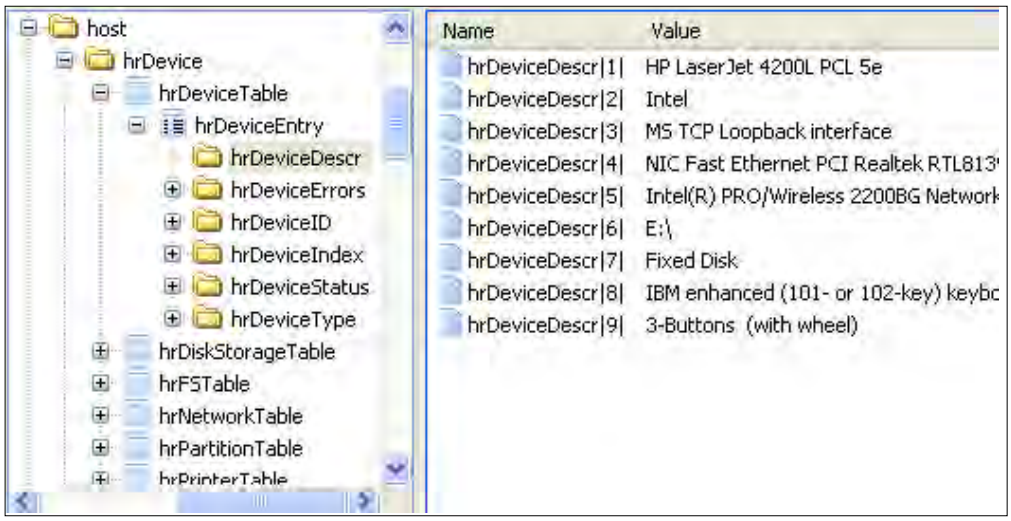
SNMP Explorer Shows the Devices Found

The example on the bottom of page 16 found seven devices with SNMP. As you can see, you can detect automatically the type of the device thanks to SNMP (for example printer or W2K host).

You can explore the information stored in every device by expanding the device tree. Every device tree seems really similar to the MIB Items tree loaded before, but unlike it, here we also have the real time information coming from the device. The MIB Items tree gives the organization of the data, while the device itself gives the data.

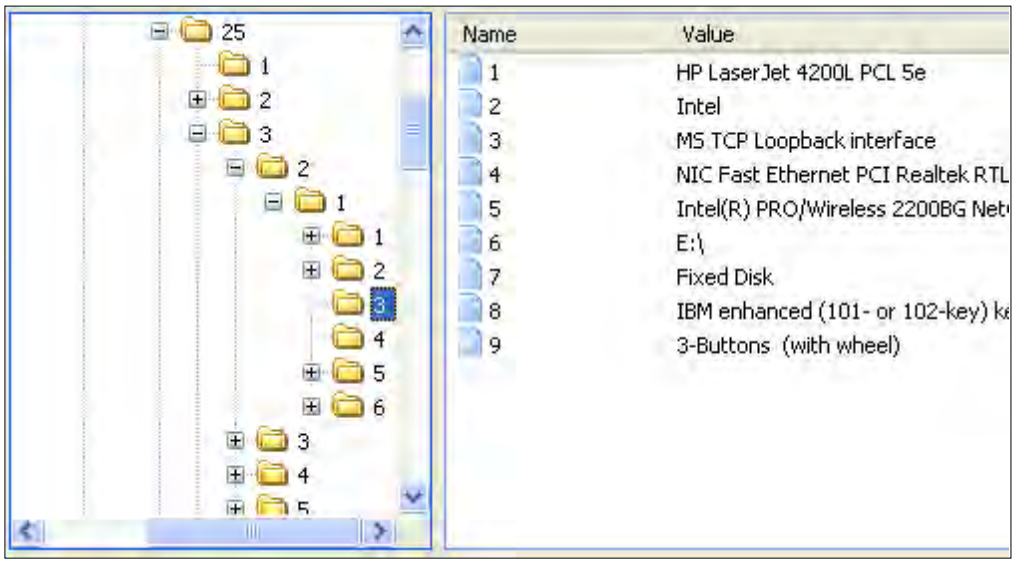
To be more precise, the **Device** tree is the **instantiation** of the MIB Items tree. When you load a MIB file into the MIB Items tree, you define the structure of a new branch of the tree that will be visible in all the devices where that branch is implemented. Without such information, the Device tree will have a really simple tree form where the branches are numbers (and it is quite impossible for a human to understand what this means).

To make the link between the MIB Items tree and the Device tree, let's make a simple example. In the figure below you will see a branch of the Device tree from SNMP Explorer that includes all the information about the devices installed on a machine.



Branch of the Tree in SNMP Explorer With Loaded MIB Items Tree

But what happens if you delete all the information from the MIB Items tree? The

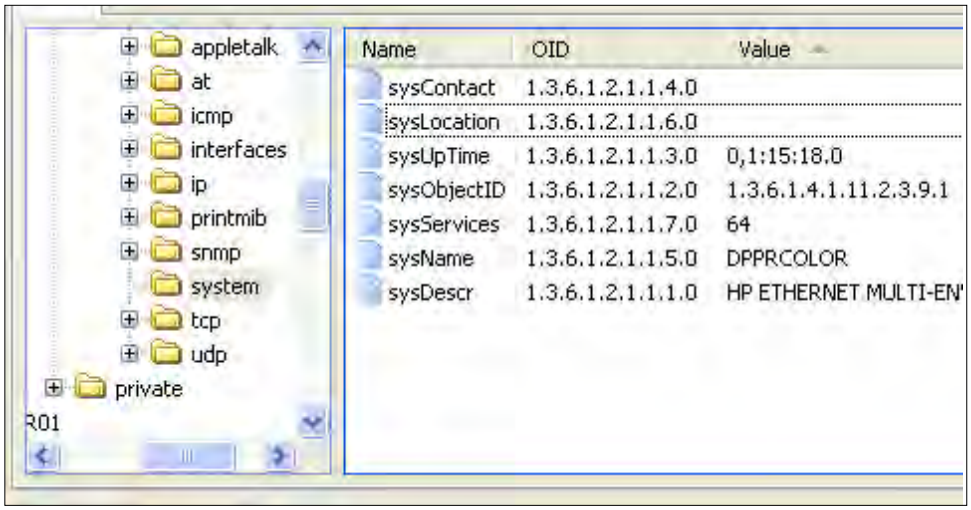


result is shown in the figure below.

Branch of the Tree in SNMP Explorer - MIB Items Tree is not Loaded

Now that we have seen why the information must be organized through the MIB Items tree, let's see how the information is organized. There are three kinds of information stored in a device: Scalar Values, Arrays and Tables.

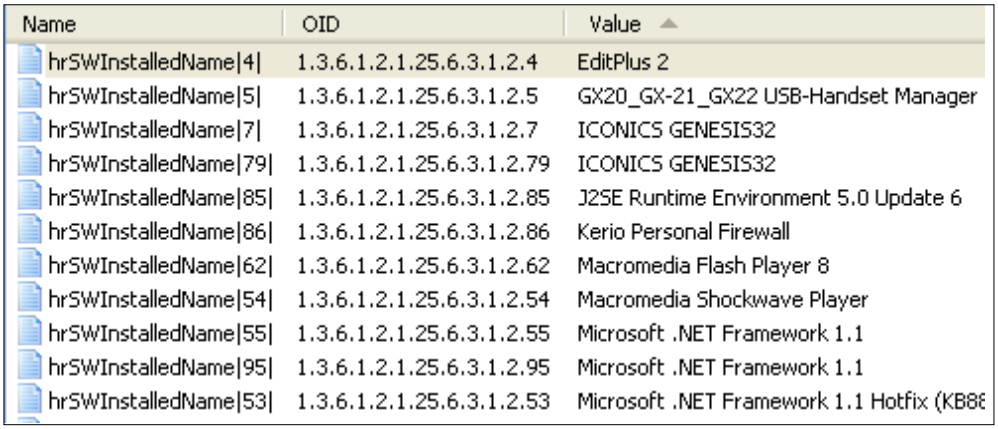
- **Scalar values:** These are the simplest type of information that can be found. Every value has its own specific name, and it is easily recognizable because its OID ends with a "*.0*' Usually these objects contain particular information about the device, such as the system uptime, the device description or the number of running processes. We can find a simple set of scalar values in the *"iso.org.dod.internet.mgmt.system"* folder, as shown in the figure below.



Scalar Values

You can add a tag connected to the scalar value by double-clicking on it. The SNMP tag will be imported into the SNMP Configurator, and it you can retrieve it through the Unified Data Browser.

- **Arrays:** If we think about information like the software installed on a machine or like the processes running, we understand that it is impossible to store them into scalar values. In fact, arrays are used to show this information. The name is the same for all the objects, but it is followed by an index that is unique to every object, and it is coming from the last part of the OID. In the figure below, for example, we have an array for all the software installed on the machine. The object *hrSWInstalledName|4|* is the fourth element of the array, and its OID ends with a "*.4*'.

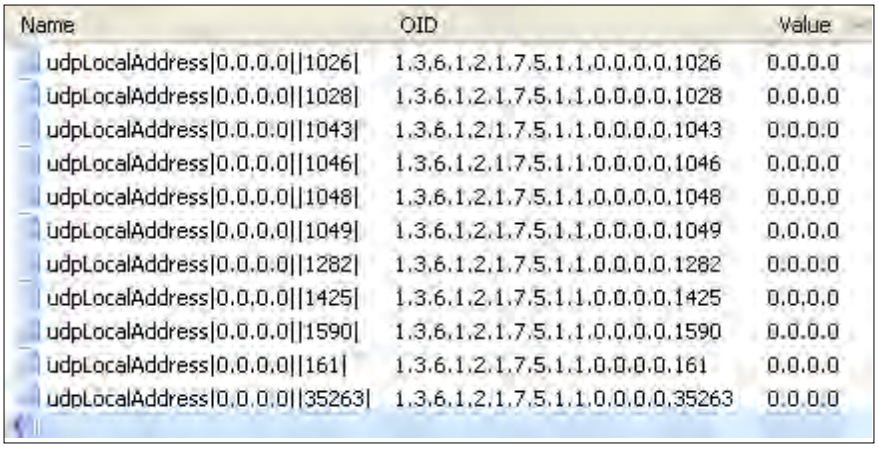


Name	OID	Value
hrSWInstalledName|4|	1.3.6.1.2.1.25.6.3.1.2.4	EditPlus 2
hrSWInstalledName|5|	1.3.6.1.2.1.25.6.3.1.2.5	GX20_GX-21_GX22 USB-Handset Manager
hrSWInstalledName|7|	1.3.6.1.2.1.25.6.3.1.2.7	ICONICS GENESIS32
hrSWInstalledName|79|	1.3.6.1.2.1.25.6.3.1.2.79	ICONICS GENESIS32
hrSWInstalledName|85|	1.3.6.1.2.1.25.6.3.1.2.85	J2SE Runtime Environment 5.0 Update 6
hrSWInstalledName|86|	1.3.6.1.2.1.25.6.3.1.2.86	Kerio Personal Firewall
hrSWInstalledName|62|	1.3.6.1.2.1.25.6.3.1.2.62	Macromedia Flash Player 8
hrSWInstalledName|54|	1.3.6.1.2.1.25.6.3.1.2.54	Macromedia Shockwave Player
hrSWInstalledName|55|	1.3.6.1.2.1.25.6.3.1.2.55	Microsoft .NET Framework 1.1
hrSWInstalledName|95|	1.3.6.1.2.1.25.6.3.1.2.95	Microsoft .NET Framework 1.1
hrSWInstalledName|53|	1.3.6.1.2.1.25.6.3.1.2.53	Microsoft .NET Framework 1.1 Hotfix (KB8

Array Values

In this case, the index of the array is an integer ("*4*'), but in SNMP the index of an array can be also composed by an IP address, a string, an integer or a combination of these objects. These indexes are called "Conceptual Rows." It is not important to understand the full meaning of this, because SNMP Explorer automatically recognizes the index of an array and shows it to you in a very user friendly way. In the example at the top of page 20, the array contains the UDP connections of the machine, and the index is composed by an IP address and an integer.

Note: As you can see, the separator for the indexes is the pipe character " | ", and this identifies the SNMP indexed data within the GENESIS32 suite.

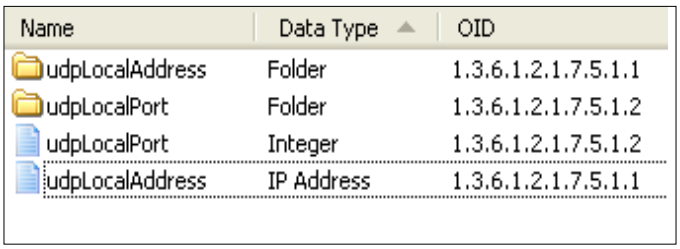


Name	OID	Value
udpLocalAddress|0.0.0.0||1026|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1026	0.0.0.0
udpLocalAddress|0.0.0.0||1028|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1028	0.0.0.0
udpLocalAddress|0.0.0.0||1043|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1043	0.0.0.0
udpLocalAddress|0.0.0.0||1046|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1046	0.0.0.0
udpLocalAddress|0.0.0.0||1048|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1048	0.0.0.0
udpLocalAddress|0.0.0.0||1049|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1049	0.0.0.0
udpLocalAddress|0.0.0.0||1282|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1282	0.0.0.0
udpLocalAddress|0.0.0.0||1425|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1425	0.0.0.0
udpLocalAddress|0.0.0.0||1590|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.1590	0.0.0.0
udpLocalAddress|0.0.0.0||161|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.161	0.0.0.0
udpLocalAddress|0.0.0.0||35263|	1.3.6.1.2.1.7.5.1.1.0.0.0.0.35263	0.0.0.0

Array Values with Complex Indexes

As you can see, the IP addresses and the integers are easily recognizable both in the name and in the end of the OID of the object (as well as a simple integer). You can add the single element of the array to the tags selected by double-clicking on it.

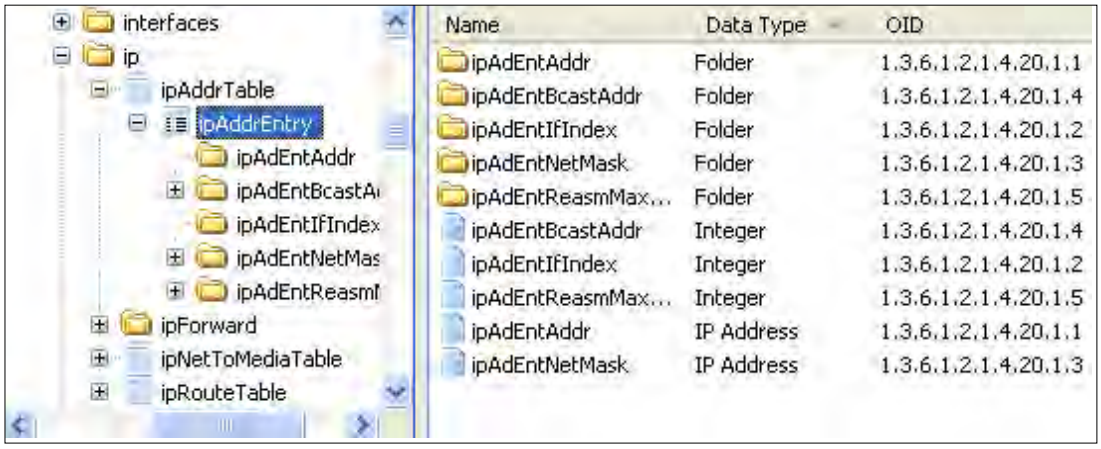
Note: The main problem about this kind of information is that part of its OID (which is the address of the information) is "hard coded" into the names. And if the addresses change, the names will change too. So if you try to retrieve the information through the old names, they will no longer be available. For example, if the UDP connection on port 1026 is closed, the tag *"udpLocalAddress|0.0.0.0||1026|"* will not be available anymore. However, the solution to this problem is really simple. The arrays are usually contained in a folder that has the same name (but without indexes). Associated with each folder is a special tag that has the same name of the folder. Selecting this tag will add the whole array to the SNMP Configurator. Then in GraphWorX32 (for example) it will be possible to select the array and manage it as a normal array.



Name	Data Type	OID
udpLocalAddress	Folder	1.3.6.1.2.1.7.5.1.1
udpLocalPort	Folder	1.3.6.1.2.1.7.5.1.2
udpLocalPort	Integer	1.3.6.1.2.1.7.5.1.2
udpLocalAddress	IP Address	1.3.6.1.2.1.7.5.1.1

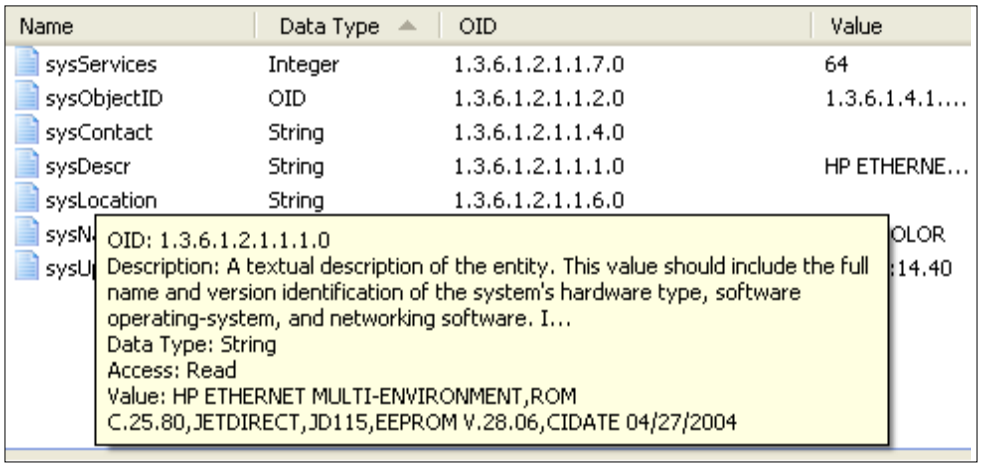
Array Tags Associated With the Arrays

- **Tables:** Tables are no more than a set of arrays. The table simply carries the information about the conceptual rows and about the type of the arrays. Every table contains a special object called "Entry" which contains the arrays. While the arrays represent the "Columns" of the table, the entry represents its "Rows." Like the arrays, there are tags associated with the tables that allow you to select the whole table.



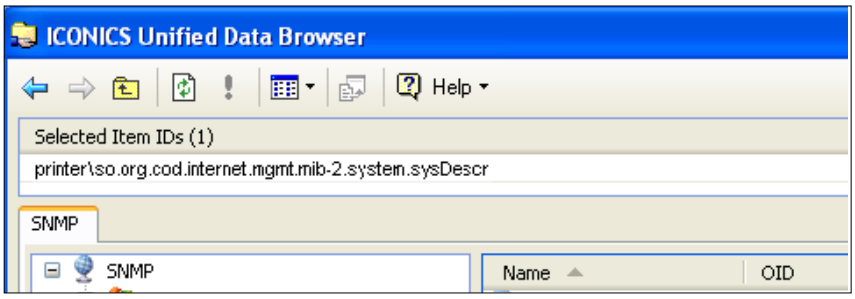
Entry Contains All the Columns of the Table, Which Are Arrays

For example, let's try to retrieve the system description from the printer. Since "*sysDescr*" is a standard information, you can try this example with every SNMP-enabled device (not just the printers). To do this, just click on the printer and then explore the tree to reach "*sysDescr*". Its path is "*iso.org.dod.internet.mgmt.mib-2.system.sysDescr*". Leaving the mouse on "*sysDescr*" shows a tooltip with all the information available about "*sysDescr*" (the information is coming from the MIB Items tree). Most of the information is also shown in the list view on the right pane.



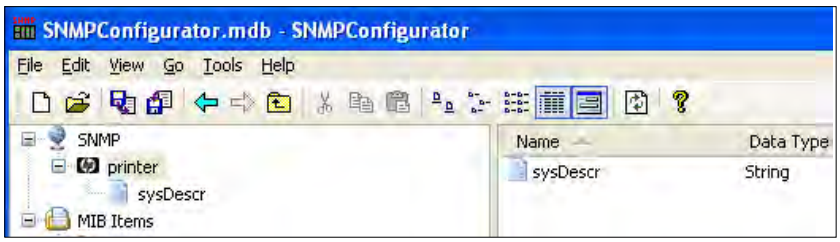
Tooltip Shows All the Information About the Object

Select the "*sysDescr*" tag by double-clicking on it. It will be displayed on the top of the SNMP Explorer. Click **OK** to send the selected tag(s) to the SNMP Configurator.



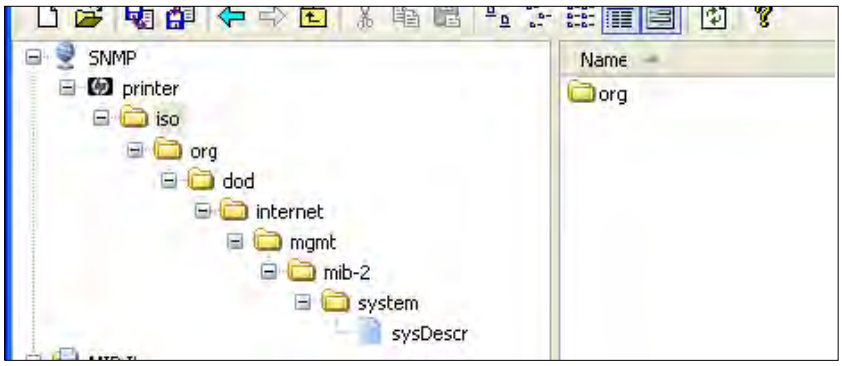
Selecting the Tag in the Unified Data Browser

Now in the SNMP Configurator the **SNMP** tree is not empty anymore. The SNMP OIDs of the device are automatically converted into GENESIS32 **SNMP tags.** These two objects are really similar (same name, same FullOID, same type...), but while the SNMP OID is stored in the device and it is unreachable by GENESIS32 applications, the SNMP tag is visible in the ICONICS Unified Data Browser and can be employed in every GENESIS32 application like a normal OPC tag.



SNMP Tags in the Configurator

There is a new device (directly imported from the SNMP Explorer) and the "*sysDescr*" item is placed under it. You can rebuild all the folders that contains sysDescr in the Explorer ("*iso*", "*org*", "*dod*", "*internet*", "*mgmt*", "*mib-2*", "*system*") by checking the **Rebuild MIB tree** check box. The result is shown in the figure below.



MIB Folder Structure Rebuilt in SNMP Tree

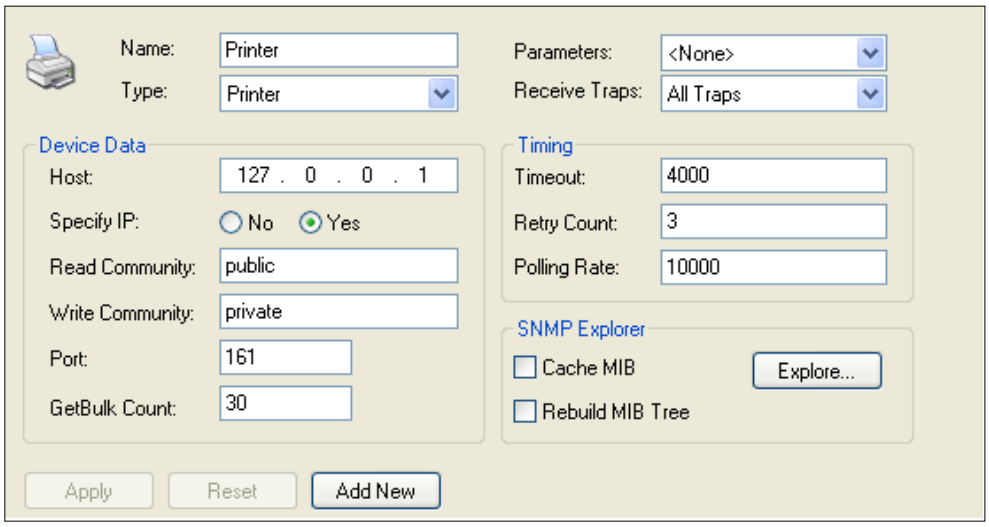
While this function is useless for one item only, it is really useful to auto-organize data when you import hundreds of OIDs from SNMP Explorer.

3.6 Organize the Collected Data

Now all the information is stored in the SNMP Configurator. The data shown under the **SNMP** tree are exactly the same data that will be shown in the ICONICS Unified Data Browser. From the SNMP Configurator, you can configure everything to make things easier in the ICONICS Unified Data Browser. Just remember that every item here is an SNMP tag, which acts like a normal OPC tag in the GENESIS32 world. So there is no problem in using SNMP tags in the same places where the OPC tags are used. Simply the source of the data is different, but this is totally transparent to you.

3.6.1 Devices

Clicking on a Device (i.e. printer) shows the device properties in the lower right pane of the SNMP Configurator.



Device Properties in SNMP Configurator

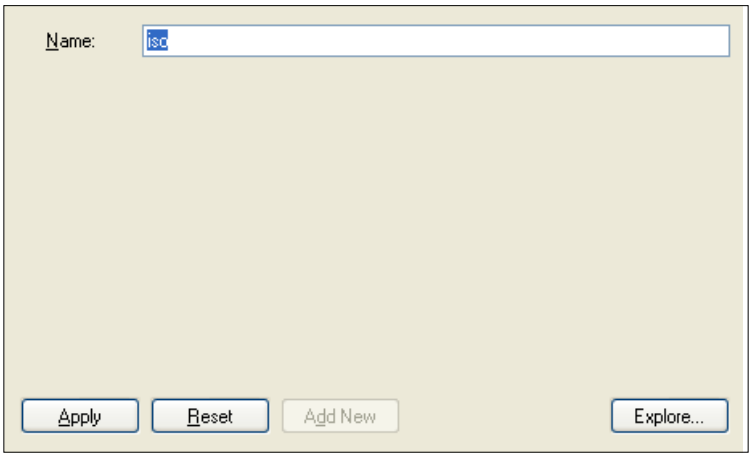
You can modify a lot of settings:

- **Name:** Specifies the name of the device, and it has nothing to do with the DNS name.
- **Type:** SNMP Explorer automatically detects the type of the device, but you can modify it among twenty different kinds of devices.
- **Host:** Specifies the IP address of the device. You can also specify the DNS Name by changing the **Specify IP** setting.
- **Read and Write Community:** These are the "passwords" that enable SNMP Connector to talk with the device. If these passwords are wrong, it would be impossible to read or write SNMP data on the device.

- **Port:** You can change the default port where the SNMP service tries to connect.
- **GetBulk Count:** SNMP GetBulk is a new SNMP message that can request a lot of information to the device at once. Thanks to this feature you can reduce the number of messages sent to the device, and as a consequence the network traffic and the device's agent working time will be lower. Here you can set the maximum number of elements that can be retrieved with a single GetBulk message. Usually GetBulk is employed for arrays and tables data retrieval.
- **Timeout:** Specifies the timeout of the SNMP requests to the device.
- **Retry Count:** This value specifies how many times the SNMP Connector must try to retrieve an SNMP value.
- **Polling Rate:** This is the period (in milliseconds) between two data requests.
- **Receive Traps:** Allows the machine to receive traps from the device (if the device is correctly configured).
- **Explore button:** You can connect directly to the selected device through the "Explore" button. The tags selected will be added directly to the device. The **Cache MIB** and **Rebuild MIB Tree** check boxes have the same functions of those seen during the Network discovery, but you can configure them differently for every device.
- **Parameters:** Allows the creation of templates that do not have a specific IP address or port. They will be specified in GraphWorX32 (for example) as a parameter of the OID. This allows users to create aliases or pre-configured displays without changing the SNMP tags every time.

3.6.2 Folders

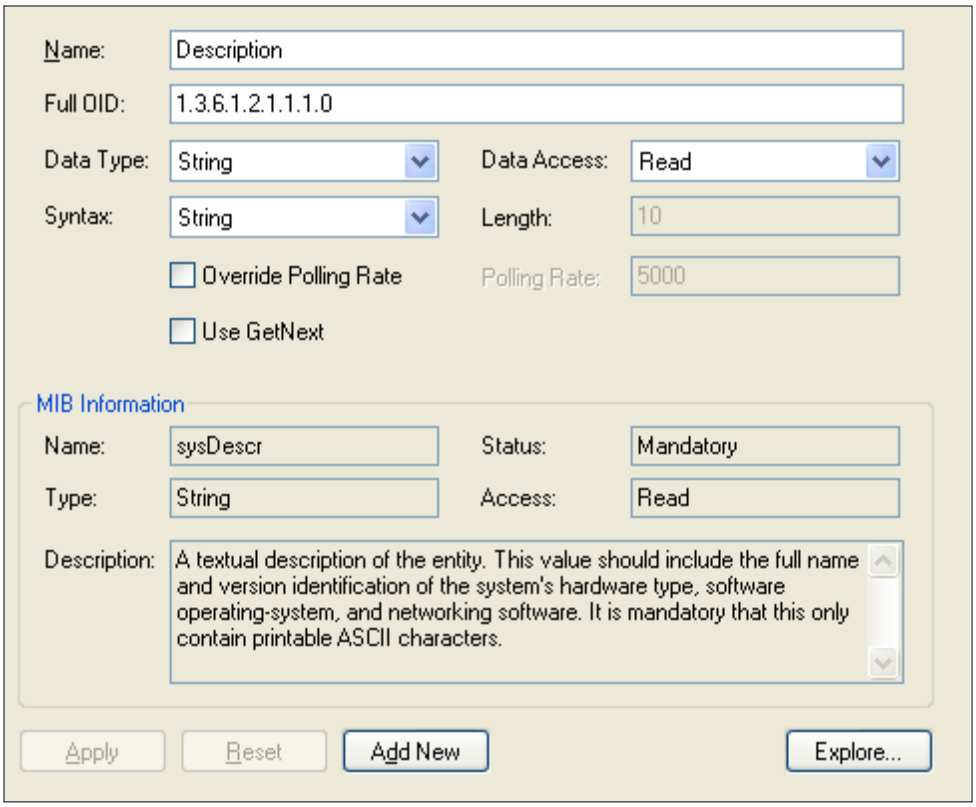
Clicking on a folder this dialog will appear. As you can see, you can only change the name and add more items/folders using the **Explore** button.



Folder Properties in SNMP Configurator

3.6.3 Items

Clicking on an item you will see this dialog.

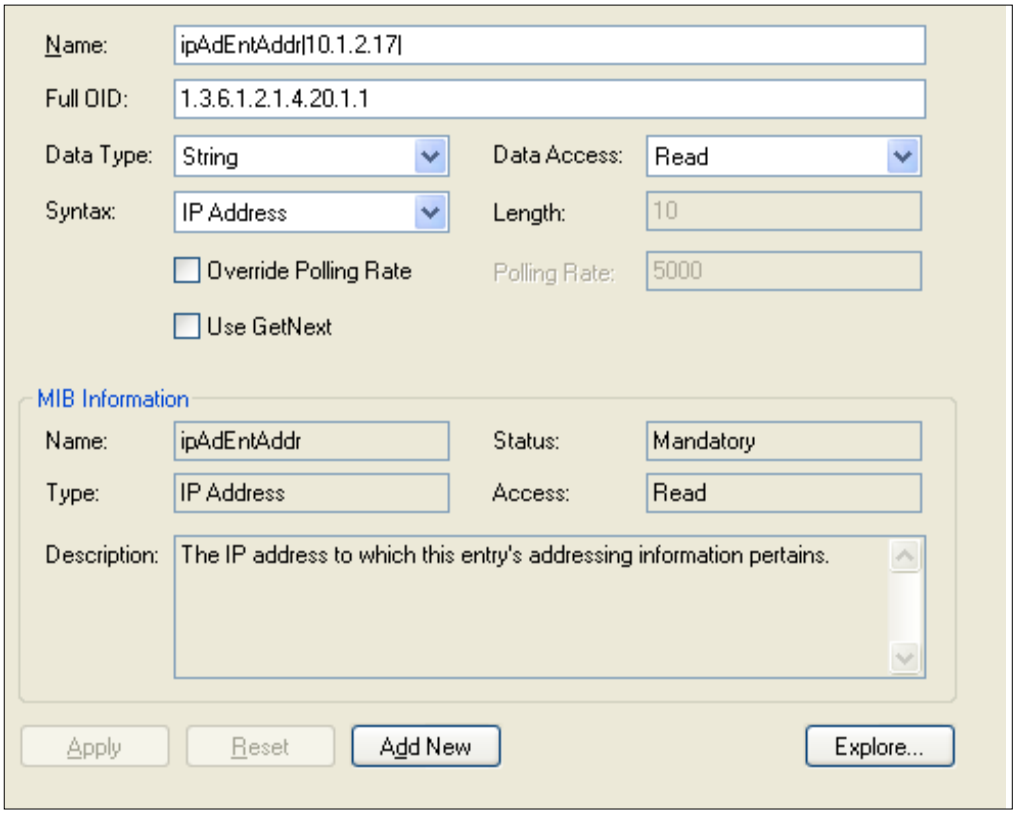


Item (Tag) Properties in SNMP Configurator

From here you can configure all the information about the SNMP tag:

- **Full OID:** The identifier for every SNMP tag is the **Full OID.** Thanks to this information we are able to retrieve the associated data in the MIB "dictionary" tree (if it exists).
- **Data Type:** Specifies the OLE data type, and is identical to the OPC data type. This field must be set to "Array" for SNMP tables.
- **Syntax:** Specifies the SNMP type of the tag. It can also be a table, but only if the SNMP item associated with the tag (which has the very same Full OID) is a table too.
- **Data Access:** A tag can be readable, writable, both or none.
- **Overriding Polling Rate:** You can override the device polling rate only for the selected tag.
- **Use GetNext:** Check this check box to retrieve values through a Get-next instead of the default Get. This field is disabled for Array Data, Table Data and writable tags.

- **Name:** The name of the tag is critical. For Scalar Values, Table Tags and Array Tags (see section 3.5, "Explore the Network") you can change without limits the name of the tags. For the array elements it is really dangerous to change the ending part of the name because it contains information about the Full OID itself. Let's see an example:

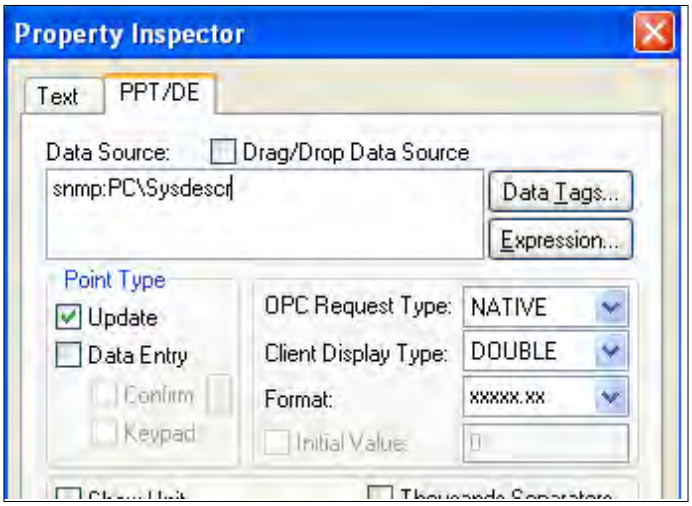


Example Tag Configuration in SNMP Configurator

In the example above the name of the tag is composed by the SNMP Item name and the index of the Array (which is the IP Address 10.1.2.17). To retrieve in runtime the value of the tag we must concatenate the Full OID "1.3.6.1.2.1.4.20.1.1" (which is the same Full OID of the SNMP Object ipAdEntAddr) and the index of the Array "10.1.2.17" to obtain the real "address" of the information: "1.3.6.1.2.1.4.20.1.1.10.1.2.17". If the index is changed or deleted, the data will not be retrievable anymore. However you can change the index coherently to obtain different elements of the array.

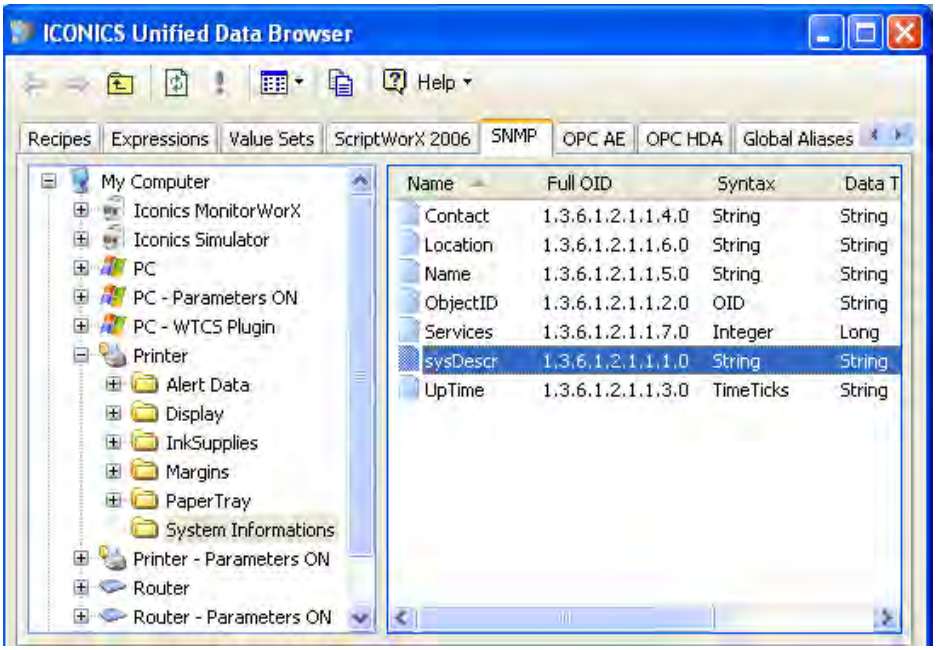
3.7 Create Your Display in GraphworX32

Now that all the SNMP tags are correctly configured, we can start GraphworX32 and click the **Process Point** button.



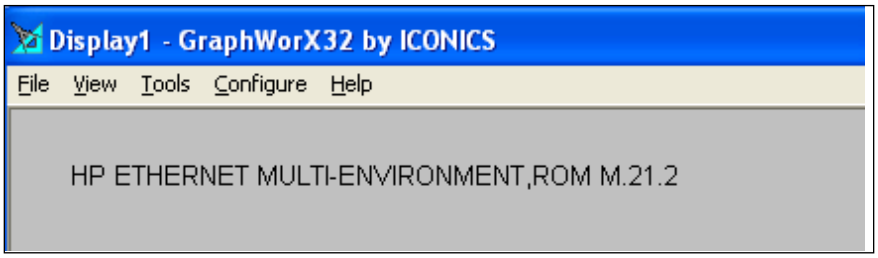
Property Inspector in GraphWorX32

Then click on **Data Tags** as if you wanted to retrieve a normal OPC tag. You will see an **SNMP** tab in ICONICS Unified Data Browser, and under **My Computer** you will see all the SNMP tags created in SNMP Configurator. It's also possible to explore other computers for SNMP tags (These computers need GenBroker configured and running. For more information, please refer to the GenBroker help documentation.)



Selecting SNMP Tags from the Unified Data Browser

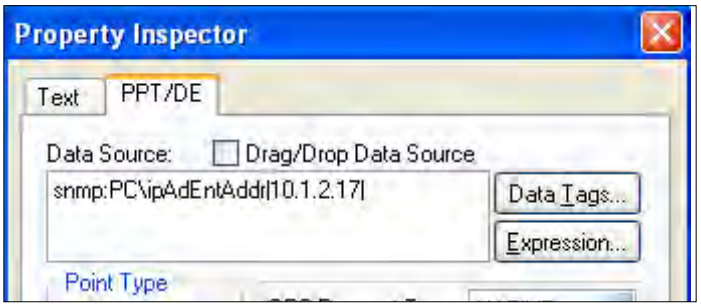
Select the "*sysDescr*" tag, for example, and click **OK** in the Property Inspector dialog. Then start runtime mode and see the system description, which is visible in the GraphWorX32 display!



SNMP Tag Resolved in GraphWorX32 Display

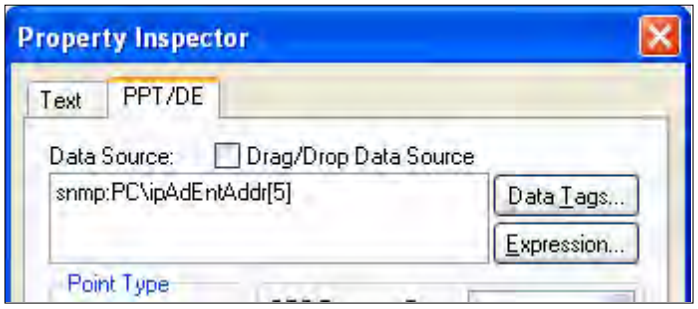
You can visualize elements of an array:

- Just by selecting the OID tag associated with the element set in the SNMP Configurator



SNMP Tag Syntax Shown in Property Inspector

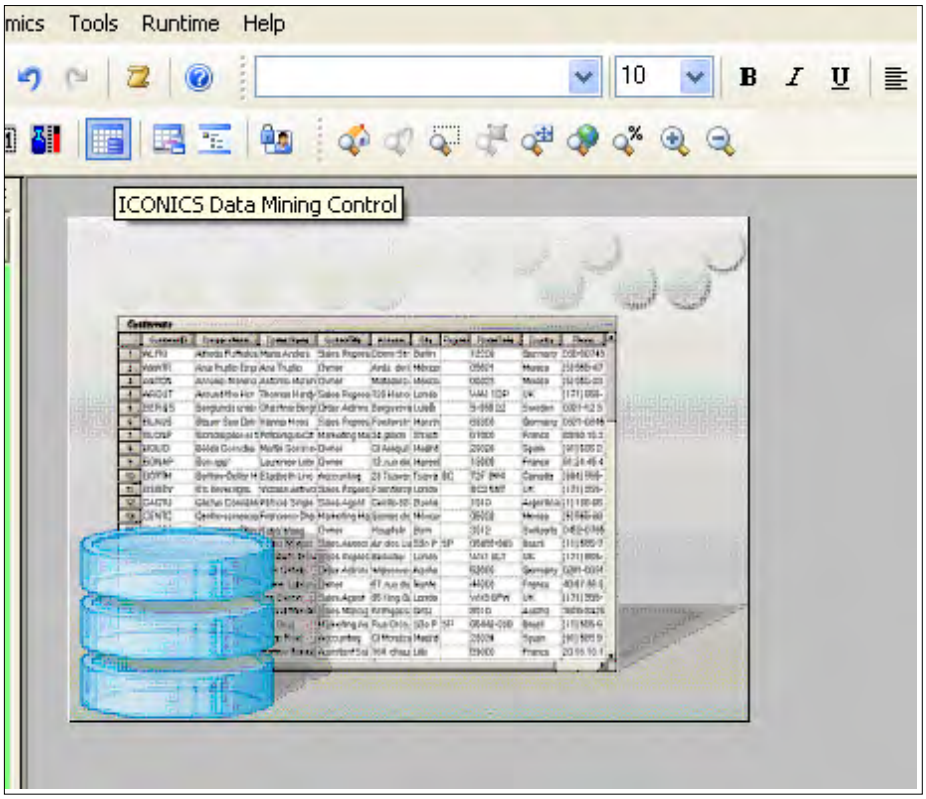
- If the whole array was imported in SNMP Configurator it's necessary to set the index of the array for the element desired (like a normal array). This is possible thanks to the native array management given by GenClient.



SNMP Tag Syntax with Index Shown in Property Inspector

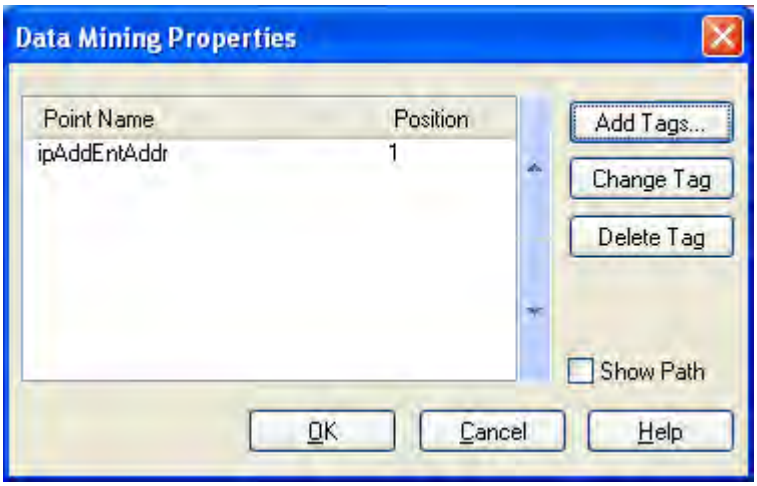
Don't be fooled by the characters! The pipe ("/") symbol is used to identify SNMP indexes (can be integers, IP addresses or even strings) and it is a part of the tag name. The square parenthesis (brackets) symbol *("[" and "]")* is given by the GENESIS32 native management of the arrays through GenClient. They are **NOT** part of the tag name, and they can be integers only. In fact, they identify the index for all generic GENESIS32 arrays, including SNMP arrays.

You can even use the **Data Mining ActiveX** to show the whole array. Just click on the Data Mining button on the ActiveX Toolbar in GraphWorX32 to open the ActiveX.



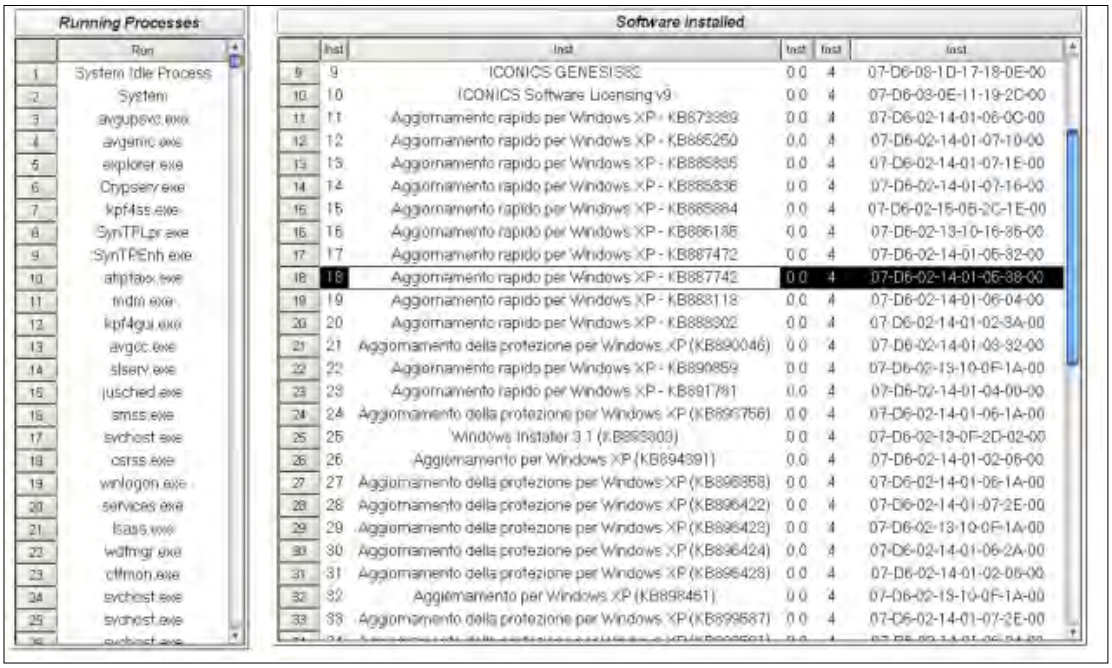
Data Mining ActiveX Control Inserted into GraphWorX32 Display

Double click on the ActiveX and select **Data Mining** in the Data Source Type. Click on **Connection Parameters** and set the Table or Array tag as the data source.



SNMP Tag Specified as Data Source in Data Mining ActiveX

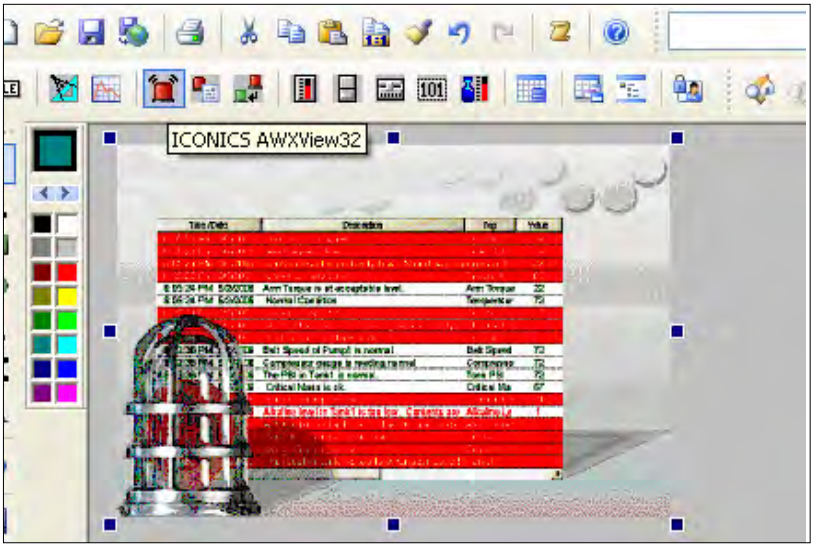
Then click on Runtime and see the ActiveX filled with the information coming from



the SNMP tag.

Data Mining ActiveX Populated with SNMP Tag Data in Runtime

Finally you can use the **AlarmWorX32 Viewer ActiveX** to visualize the traps. To do this click on the AlarmWorX32 Viewer ActiveX button on the ActiveX Toolbar in GraphWorX32. If the SNMP environment is configured properly (see Section 2.6), no further work is needed to show the traps through the AlarmWorX32 Viewer ActiveX.



AlarmWorX32 Viewer ActiveX

4 SNMP Simulator

4.1 SNMP Agents Basics

An SNMP agent is a small piece of software that is installed on a device (computer, printer, router, etc.) and that provides some data about that device. These data can be accessed thanks to an SNMP Manager, such as the ICONICS SNMP Configurator.

4.2 ICONICS SNMP Simulator features

The ICONICS SNMP Simulator generates several SNMP variables that can be used to evaluate the usage of the SNMP protocol in a test environment.

When using the ICONICS SNMP Configurator to browse a device running the ICONICS SNMP Simulator, we will generate several SNMP requests to get data from the agent. The agent relies on the Microsoft SNMP Service, which must be installed and running to make it work. Please refer to Section 2.5 of this document to install SNMP on your computer.

The variables that the ICONICS SNMP Simulator generates are organized in several categories. Let's analyze them to explore the features that this agent implements.

- **Configuration section:** Thanks to these SNMP variables, you can customize the behavior of the agent, depending on your needs.
- **Host section:** These SNMP variables simulate some typical host values, such as CPU temperature, memory usage, etc.
- **Network section:** Here we have some basic network simulated values: packets sent and received, bytes in and out, and packet loss.
- **Printer section:** This is a simulated printer that will allow you to print, refill ink and paper, and analyze problems of this pure virtual printer.
- **Router section:** A simulated router with some typical values.
- **Signals section:** Some basic signal, such as sine, square, random, and ramp. These are the same signals that you can also find in the OPC Simulator.

These are the main features of the ICONICS SNMP Simulator. Using these simulated variables and device, you can simulate a local network that you want to monitor, you can build your own GraphWorX32 display, which you can later link to some real data, coming from real hosts and devices.

4.3 Using ICONICS SNMP Simulator

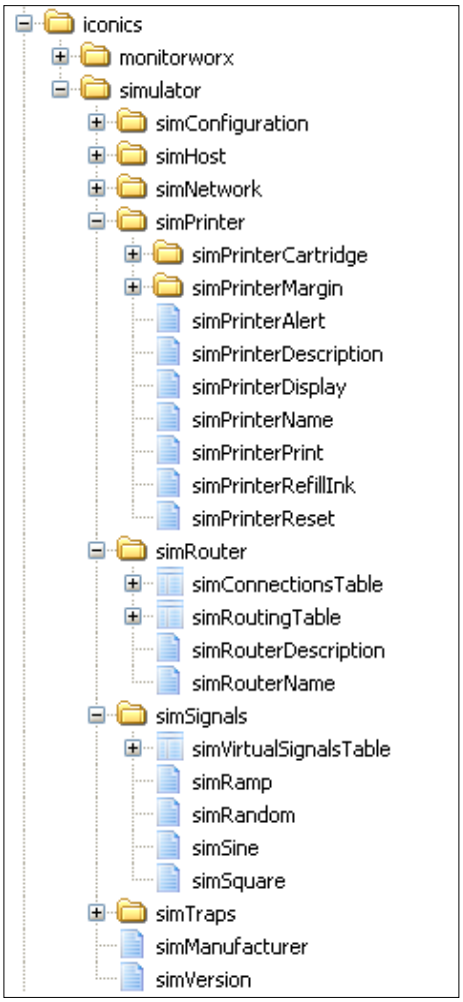
In order to use the data exposed by the ICONICS SNMP Simulator, you need to use the ICONICS SNMP Configurator to choose the variables to use, as we already discussed in Sections 3.2 and Section 3.6. Once you have configured the

variables you need, you can simply create your GENESIS32 applications as you can do with other SNMP agents and OPC servers.

The Simulator also sends some sample traps that can be captured and displayed as if they were other kinds of alarms.

To use it most effectively, we must analyze the MIB that the SNMP Simulator implements.

The Simulator MIB contains a couple of OIDs and several branches. The OIDs simply contain the name of the manufacturer and the Simulator version. The most interesting data are contained in the branches, which need a detailed description in order to be used most effectively. We will analyze them, one by one.



- **simHost branch:** The simHost branch contains several simulated variables that describe some characteristics of a device. We have OIDs that contain the CPU Usage percentage and temperature. We also have the CPU fan speed. If you increment this speed, you will notice that the CPU temperature will decrease, due to better cooling. The same also happens for the motherboard. We have again temperature and fan speed for it. The memory usage can also be simulated. All the described OIDs will change at runtime thanks to some random policy. We also have two static OIDs (they do not change during time) that are the host name and host description.
- **simNetwork branch:** This branch contains some OIDs that can be used to simulate some basic network values. For example you can use them to show the number of packets coming in and out from an interface, as well as the bytes sent and received. The packet loss number can also be simulated. Its value can be controlled thanks to a parameter, whose description will be found in the simConfiguration branch.
- **simPrinter branch:** In this branch we have a large number of OIDs that can be used to simulate a printer. You will be able to perform several operations, such as printing, refilling paper and ink, and so on. In the main

branch we have the printer name, a description, and probably most important, the printer display. In this OID we will have the content of the display, which will change depending on the current printer task and status. An alert OID is used to check the alert status of the printer, to understand if it is working properly, or if we have some problem that has to be fixed (i.e. missing paper). There are also four read-writable OIDs that are used to send some command to the printer. Writing a value to these variables has the side effect of having the printer printing, resetting itself, and refilling ink and paper.

We also have some sub-branches: **simPrinterCartridge** contains current ink level and maximum ink level. Use this to check how much ink you still have in your toner cartridge. You can check for it to be over a minimum level, otherwise you can refill it. Take a look at the printer display while the ink level is low. In **simPrinterMargin** we have the values of the paper margin that are currently used. Feel free to change these. **simPrinterTray** tells how much paper you have in the paper tray, as well as the maximum number of pages you can load in it. As well as for the cartridge, you can check for low paper level. The last branch we have is **simPrinterCfg,** which allows you to configure the printer speed in pages per minute. A higher number here makes your simulated printer print faster. Modify this value to fit your simulation needs.

- **simRouter branch:** The simRouter main branch contains two OIDs and two tables. The OIDs are a simple name and description variables, while the tables are sub-branches containing information about the connections and routing tables. The connections table contains four columns that are source ip/port and destination ip/port. The number of elements in the connections table can be configured, but this is explained later, in the simConfiguration section. The routing table contains three columns that are the destination IP address, the network mask, and the next hop where to send data, in order to reach the specified destination. The number of elements in this table can be configured as well.

- **simSignals branch:** This branch contains four classic signals: ramp, random, sine and square. These are like the signals you can find in the OPC Simulator, but are retrieved and generated by the SNMP Simulator and the SNMP Service. The ramp signal increments every second. The sine signal ranges between -1000 and 1000 in a fixed amount of time. The square signal changes between 0 and 1 every few seconds. The random signal ranges randomly between 0 and 1000.

  The **simVirtualSignalsTable** is a table that contains a column for each of the four signals listed above: ramp, random, sine and square. Thanks to this table you can generate a large number of signals, modifying some configuration values. Please note that except for random signals, the others will have very similar values, due to their nature. This table is only intended to have a large amount of data to test the use of tables and arrays, disregarding the signals' values. If you need to have some different variable values, please use the random signals.

- **simConfiguration branch:** This branch contains several OIDs that are used to tune the ICONICS SNMP Simulator behavior. These are very important to make a correct use of the Simulator, so we give a detailed description for each of them:
  - **simCfgConnectionsCount:** Setting this variable will modify the number of elements in the connections table, under the **simRouter** branch.
  - **simCfgMaxBytesPerSec:** This one specifies the maximum number of bytes that can be received every second, in the **simNetwork** branch.
  - **simCfgMaxPacketsPerSec:** This is as the previous one, but affects the number of packets rather than the bytes.
  - **simCfgPacketLoss:** This one represents the average number of packets every which we have a packet loss. For example, a value of 1000 means that once every thousand packets we can have a packet loss. Since the loss is randomly generated, you are not guaranteed to have "exactly" one packet lost every **simCfgPacketPerSec** packets.
  - **simCfgRoutingCount:** This is the number of elements in the routing table in **simRouter** branch. Try to modify this to have the number of elements changing at runtime.
  - **simCfgSignalCount:** This is the number of elements in the **simVirtualSignalsTable** in the **simSignals** branch.
  - **simCfgSignalReset:** When you write any value to this OID, the effect is of resetting all the signals in the **simSignals** branch.
  - **simCfgTemperature:** This OID can be only set to 'C', 'F' or 'K'; otherwise it will be set by default at 'C'. The three allowed values determine if the temperature values used in the Simulator will be displayed in Celsius, Fahrenheit or Kelvin degrees.
  - **simCfgTrapFlag:** This is a flag that enables/disables the trap sending by the Simulator. You can start/stop them by simply writing 1 or 0 to this OID.
  - **simCfgTrapTimer:** This is the trap timer in milliseconds. A value of 2000 means that the ICONICS SNMP Simulator will send a trap every two seconds. Choose your favorite value.

Not all the values are allowed for these variables. You may notice that you cannot write some values to a variable, because the Simulator will force it to a minimum or maximum value. Feel free to try to modify all the values as you want, to test the Simulator. Please note that all the changes will last until next SNMP Service restart (i.e.: until reboot) because they are kept in memory only. The changes are not lost if you close the ICONICS SNMP Simulator, or the GraphWorX32 displays where you use the ICONICS SNMP Simulator.

5 Conclusions

5.1 What SNMP Connector Can Offer

In this document we have seen how to quickly analyze a network, discover new devices and create SNMP tags. However, this is just a little part of what is possible to do with SNMP Connector. Searching through the MIBs of the devices, you can find every kind of information, and we have already done a simple template database to show the main information of the most common devices. Moreover, within the SNMP Connector there are also the SNMP Simulator and the SNMP MonitorWorX agent, which are able to display simulated analog data (i.e. the motherboard fan speed or the ramp signal) and the MonitorWorX data through SNMP, making the data available to all the SNMP managers in the world. This proves that the only limit to what is possible to do with SNMP is just the imagination.

ICONICS Support Service
support@iconics.com

Founded in 1986, ICONICS is an award-winning independent software developer offering real-time visualization, HMI/SCADA, energy, fault detection, manufacturing intelligence, MES and a suite of analytics solutions for operational excellence. ICONICS solutions are installed in 70% of the Fortune 500 companies around the world, helping customers to be more profitable, agile and efficient, to improve quality and be more sustainable.

ICONICS is leading the way in cloud-based solutions with its HMI/SCADA, analytics, mobile and data historian to help its customers embrace the Internet of Things (IoT). ICONICS products are used in manufacturing, building automation, oil & gas, renewable energy, utilities, water/wastewater, pharmaceuticals, automotive and many other industries. ICONICS' advanced visualization, productivity, and sustainability solutions are built on its flagship products: GENESIS64™ HMI/SCADA, Hyper Historian™ plant historian, AnalytiX® solution suite and MobileHMI™ mobile apps. Delivering information anytime, anywhere, ICONICS' solutions scale from the smallest standalone embedded projects to the largest enterprise applications.

ICONICS promotes an international culture of innovation, creativity and excellence in product design, development, technical support, training, sales and consulting services for end users, systems integrators, OEMs and Channel Partners. ICONICS has over 300,000 applications installed in multiple industries worldwide.

World Headquarters
100 Foxborough Blvd.
Foxborough, MA, USA, 02035
Tel: 508 543 8600
Email: us@iconics.com
Web: www.iconics.com

European Headquarters
Netherlands
Tel: 31 252 228 588
Email: holland@iconics.com

Czech Republic
Tel: 420 377 183 420
Email: czech@iconics.com

France
Tel: 33 4 50 19 11 80
Email: france@iconics.com

China
Tel: 86 10 8494 2570
Email: china@iconics.com

Italy
Tel: 39 010 46 0626
Email: italy@iconics.com

UK
Tel: 44 1384 246 700
Email: uk@iconics.com

India
Tel: 91 22 67291029
Email: india@iconics.com

Germany
Tel: 49 2241 16 508 0
Email: germany@iconics.com

Australia
Tel: 61 2 9605 1333
Email: australia@iconics.com

Middle East
Tel: 966 540 881 264
Email: middleeast@iconics.com

Microsoft Partner
Gold Application Development

Microsoft Partner
2014 Partner of the Year Winner
Public Sector: CityNext

64™ · Windows 8 Compatible · Windows 10 Compatible · Windows Server 2012 Certified · BTL · CERTIFIED OPC · vmware READY

VISUALIZE · ANALYZE · MOBILIZE · CLOUD

www.iconics.com