

# Technical Note SNMP Interface

## Introduction

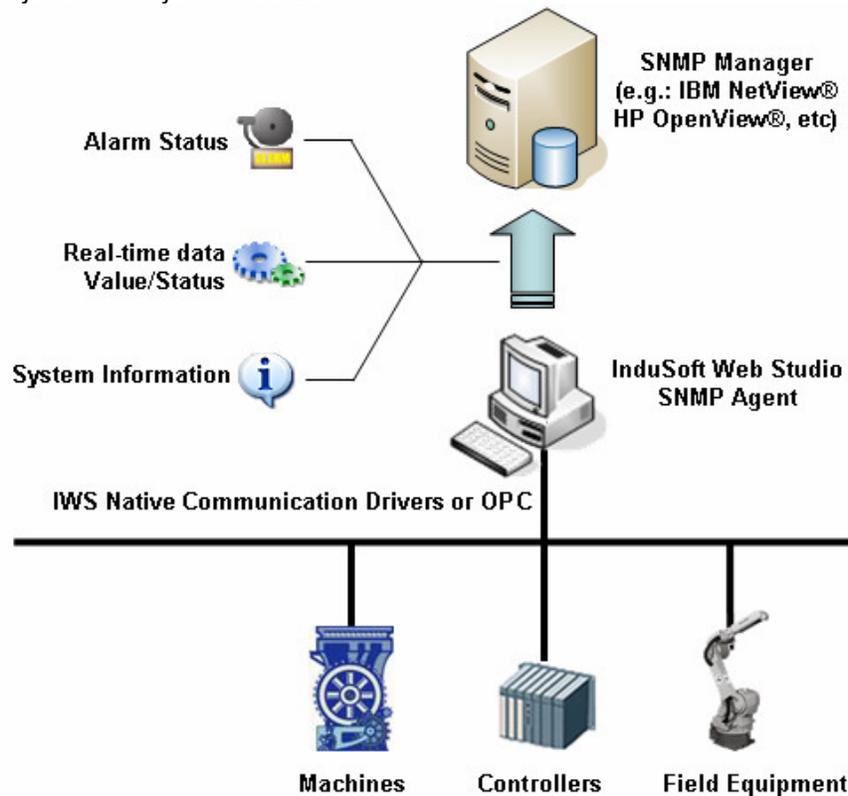
This document describes the SNMP Interface provided with InduSoft Web Studio (IWS) and CEView.

Simple Network Management Protocol (SNMP) is a popular protocol for network management. It is used for collecting information from, and configuring, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP) network.

In a typical SNMP architecture, there is one SNMP Manager station which Status Information from Devices Running SNMP Agent.

Using IWS or CEView, you can provide real-time data from the devices connected to the SCADA/HMI system through the native communication drivers (or OPC interface) – controllers, machines, and so forth – to a third-party SNMP Manager, such as HP Openview, Novell NMS, IBM NetView, or Sun Net Manager. In other words, IWS can work as a gateway between the devices in the plant-floor and the corporate equipment management system (SNMP Manager) in the ERP.

Furthermore, IWS can also request data from third-party SNMP Agents and display this information through its graphical interface, locally or remotely on the Thin Client stations.



## Installation on Windows CE

### Install Windows SNMP Service:

The SNMP Server must be previously enabled in the image of the Windows CE device. If your Windows CE device does not provide the SNMP Agent, please consult your hardware vendor and ask them to add this service to the image of the device.

### Install Studio SNMP Agent files:

Copy the following files into a non-volatile directory of the Windows CE device:

- StudioSnmp.dll
- StudioToolkit.dll
- StudioSnmpReg.exe

For example, the files could be copied into a directory like \HardDisk\StudioSnmp\ or \Storage Card\StudioSnmp\. The exact name of the non-volatile directories varies, depending on Windows CE device being used. It is recommended that these files are not stored at the same directory as CEView.exe.

**Note:** The files mentioned above are stored in the following sub-folder after installing InduSoft Web Studio in your computer: \Redist\<OS\_Version>\<ProcessorType> (e.g.: C:\Program Files\InduSoft Web Studio v6.1\Redist\Wince 4.0\x86). You can use the Send Files command from the Execution Environment dialog of InduSoft Web Studio to download these files to your Windows CE device.

### Configure Studio SNMP Agent:

The file StudioSnmp.dll is a SNMP extension agent. It must be registered to allow the Windows SNMP Service to use it. To register the agent, execute the tool StudioSnmpReg.exe. It will display a dialog with two tabs “Agent” and “Traps”.

The tab “Agent” contains 5 fields:



1) Status

It indicates if the Studio SNMP Agent is already registered. It can be “Registered” or “Not registered”. At the first time you execute this tool, status is “Not registered” because the agent is not registered yet. Change the value to “Registered” to register it.

2) Path

The full path to the file StudioSnmp.dll.

3) TCP Port

It is the TCP port used by the Studio SNMP Agent to communicate with Studio applications. Default value is 1234. If the Studio application uses another port, this field must be changed accordingly.

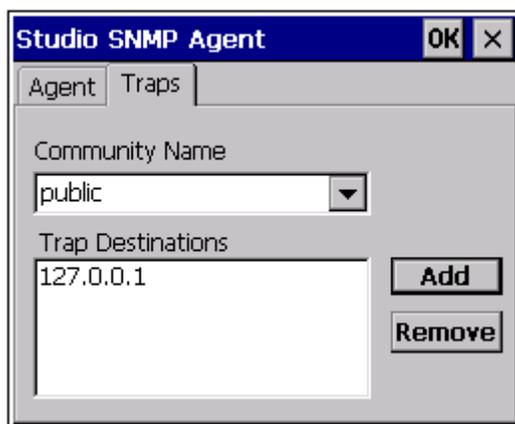
4) Enterprise Number

Unique number, which defines the position in the SNMP tree where the Studio data will be exposed (.iso.org.dod.internet.private.enterprise.[Enterprise Number]). By default, the Enterprise Number is 26640.

5) Trap Interval

It is the period, in seconds, between traps. Default value is 2. If the value is set to zero, no traps will be generated.

The tab “Traps” contains the list of machines that can receive trap messages. If you don’t plan to use traps in your application, it is not necessary to configure anything in this tab.



1) Community Name

Contains the list of communities already registered in the device.

2) Trap Destinations

It contains the list of trap destinations for the selected community. Trap destinations can be IP addresses or host names. Use the buttons “Add” and “Remove” to edit the list.

After the configuration is done, click the “OK” button to save it. Then reboot the device.

## Configuring an Application

The Studio SNMP Agent exposes data from the current application via the SNMP protocol. Data is exposed as a sub tree with root in .iso.org.dod.internet.private.enterprise.[Enterprise Number], where [Enterprise Number] is the number configured in the field “Enterprise Number” of the configuration tool.

The structure of the sub tree and the definition of the data to be exported are made in the file SNMP.INI. This file must be created in the directory “Config” of the application.

Three kinds of objects can be configured:

- Scalar values
- Tables
- Traps

### 1) Scalar values

A Scalar value exposes the value of a single Studio tag as an SNMP object.

```
[<Node Name>]
item = scalar
oid = <relative oid>
type = string | integer | unsigned32
access = read-only | read-write
tag = <tag name>
```

<b>Node Name</b>	Name of the node. It can be any string up to 64 characters. This name must be unique; it cannot be reused by other objects.
<b>oid</b>	Relative object identifier. This OID is appended to the root OID to make the real OID.
<b>type</b>	The type of data, as seen in the SNMP tree. It can be “string”, “integer” or “unsigned32”.
<b>access</b>	“read-only” or “read-write” When access=“read-write”, the tag can be changed by a SNMP PUT request.
<b>tag</b>	The name of the tag.

Example:

```
[scalar1]
OID = 1.2
item = scalar
type = string
access = read-only
tag = time
```

This example will generate a SNMP object with OID .iso.org.dod.internet.private.enterprise.[Enterprise Number].1.2. This object will return the value of the tag “Time” typed as string.

## 2) Tables

A table exposes one or more arrays of Studio tags as SNMP tables.

```
[<Node Name>]
item = table
oid = <relative oid>
Columns = <number of columns>
Column[<column number>].tag = day
Column[<column number>].type = Integer
Column[<column number>].access = read-only
Indexes = <number of tags>
Index[<index part>].tag = Table1.Col2
Index[<index part>].type = string | integer | unsigned32
```

<b>Node Name</b>	Name of the node. It can be any string up to 64 characters. This name must be unique; it cannot be reused by other objects.
<b>oid</b>	Relative object identifier. This OID is appended to the root OID to make the real OID for the table.
<b>columns</b>	Number of columns
<b>column[n].tag</b>	The name of the tag array displayed in column n. First column is column n=1.
<b>column[n].type</b>	The type of data in column n.
<b>column[n].access</b>	“read-only” or “read-write” When access=“read-write”, the tag can be changed by a SNMP PUT request.
<b>indexes</b>	Optional. Number of tags used to compose the index.
<b>Index[n].tag</b>	Optional. The name of the tag array used to compose the index.
<b>Index[n].type</b>	Optional. The type of data.

The number of rows is equal to the size of the smallest array that is used as column or index.

Example:

```
[table1]
OID = 1.3
item = table
Columns = 3
Column[1].tag = Col1
Column[1].type = integer
Column[1].access = read-only
Column[2].tag = Col2
Column[2].type = integer
Column[2].access = read-write
Column[3].tag = Col3
Column[3].type = string
Column[3].access = read-write
Indexes = 1
Index[1].tag = Col1
Index[1].type = integer
```

This example uses three tag arrays, “col1”, “col2” and “col3”, to create an SNMP table. The table is indexed by “col1”. The OID of the table is .iso.org.dod.internet.private.enterprise.[Enterprise Number].1.3.

### 3) Traps

Defines traps generated by Studio.

```
[<Node Name>]
item = trap
oid = <relative oid>
trigger = <tag name>
Columns = <number of columns>
Column[<column number>].oid = <relative oid>
Column[<column number>].tag = <tag name>
Column[<column number>].type = string | integer | unsigned32
```

<b>Node Name</b>	Name of the node. It can be any string up to 64 characters. This name must be unique; it cannot be reused by other objects.
<b>oid</b>	Relative object identifier. This OID is appended to the root OID to make the real OID for the table.
<b>trigger</b>	Name of the tag that triggers the trap. Every time the value of this tag changes, a new trap is generated;
<b>columns</b>	Number of values returned by the trap.
<b>column[n].tag</b>	The name of the tag returned as value n.
<b>column[n].type</b>	The type of data in returned as value n.
<b>column[n].oid</b>	OID of data in returned as value n.

#### Example

```
[trap1]
item = trap
OID = 1.4
trigger = second
Columns = 2
Column[1].tag = date
Column[1].type = string
Column[1].oid = 1.5.1
Column[2].tag = time
Column[2].type = String
Column[2].oid = 1.5.2
```

This example configures a trap that is generated every time the tag “second” changes. It returns 2 string values, from the tags “date” and “time”.

## Troubleshooting

**Problem:** no information appears at .iso.org.dod.internet.private.enterprise.[Enterprise Number]

- Check if Studio is running. If Studio is not running, the SNMP agent exposes no information.
- Check the TCP port. The TCP port used by the Studio TCP Server in the current application must be the same as the one informed during the agent configuration.
- Check if the file SNMP.INI is in the Config directory of the current application.
- Maybe the SNMP Service is not able to load the extension agent. The SNMP Service loads the extension agent StudioSnmp.dll during Windows initialization process. For some hardware devices or some Windows CE images, it is possible that the non-volatile directory is mounted only after the SNMP service starts. In this case, the SNMP service cannot load the agent file in startup, and the Studio SNMP Agent will not work. A possible workaround for this situation is creating a new Window CE image that includes the files StudioSnmp.dll and StudioToolkit.dll in the \Windows\ directory.

## Map of Revisions

Revision	Author	Date	Comments
A	Eduardo Castro	March 24th, 2008	Initial revision
B	Lourenço Teodoro	October 30th, 2011	Migrated from the CSV to the INI