Introduction

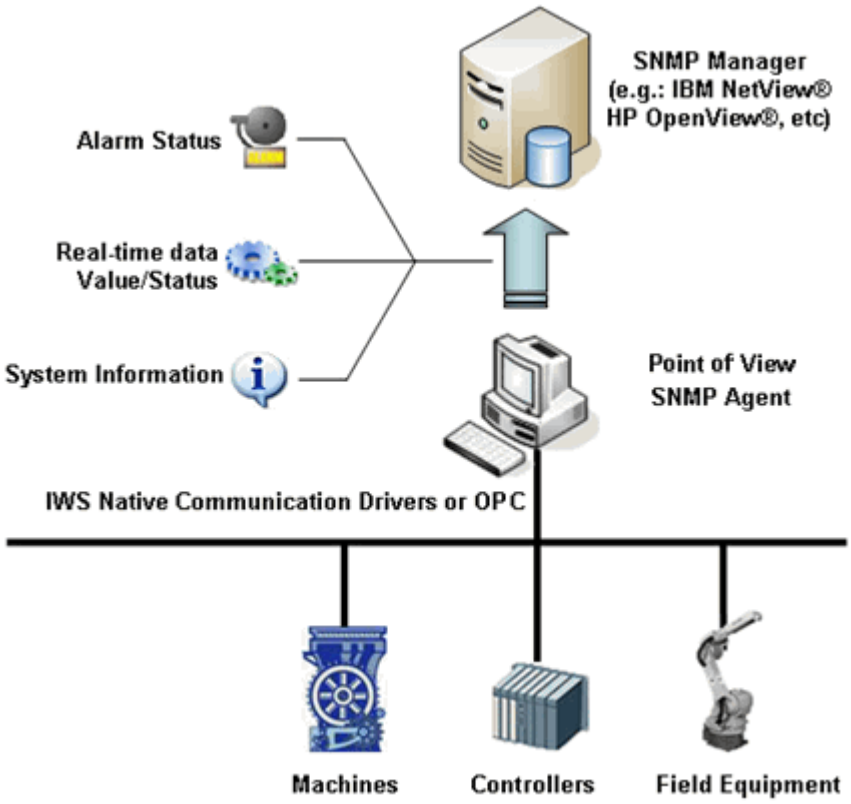
This document describes the SNMP Interface provided with Point of View (POV).

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 POV, 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, POV can work as a gateway between the devices in the plant-floor and the corporative equipment management system (SNMP Manager) in the ERP.

Furthermore, POV 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.



Configuring an Application

The POV 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 POV tag as an SNMP object.

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

Node Name	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.
oid	Relative object identifier. This OID is appended to the root OID to make the real OID.
type	The type of data, as seen in the SNMP tree. It can be “string”, “integer” or “unsigned32”.
access	“read-only” or “read-write” When access=“read-write”, the tag can be changed by a SNMP PUT request.
tag	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 POV 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
```

Node Name	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.
oid	Relative object identifier. This OID is appended to the root OID to make the real OID for the table.
columns	Number of columns
column[n].tag	The name of the tag array displayed in column n. First column is column n=1.
column[n].type	The type of data in column n.
column[n].access	“read-only” or “read-write” When access=“read-write”, the tag can be changed by a SNMP PUT request.
indexes	Optional. Number of tags used to compose the index.
Index[n].tag	Optional. The name of the tag array used to compose the index.
Index[n].type	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 POV.

```
[<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
```

Node Name	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.
oid	Relative object identifier. This OID is appended to the root OID to make the real OID for the table.
trigger	Name of the tag that triggers the trap. Every time the value of this tag changes, a new trap is generated;
columns	Number of values returned by the trap.
column[n].tag	The name of the tag returned as value n.
column[n].type	The type of data in returned as value n.
column[n].oid	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 is POV is running. If POV is not running, the SNMP agent exposes no information.
- Check the TCP port. The TCP port used by the POV 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, 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 POV SNMP Agent will not work.