PowerSNMP for ActiveX Upgrade Guide

The very successful "PowerSNMP for .NET" product was used as our model for this new ActiveX version that adds the following features:

- SNMP version 3 security
- IPv6 addressing
- Improved Table retrieval and formatting
- New 64-bit version
- Dart.CoreAx.dll is introduced, offering classes for use by Dart.SnmpAx.dll (*marked with asterisks below)

This guide walks the reader through the changes required to upgrade an existing PowerSNMP for ActiveX project to use these new features. All product GUIDs are new, so previously licensed products can be installed 'side-by-side' with the new product to make this process easier.

New Vs Old

Considering the added capabilities and reliability goals, our designers decided to model this upgrade after the seasoned PowerSNMP for .NET product (instead of simply extending the previous design). This had the desired result of better code reusability/reliability, fewer properties and methods, and enhanced usability. We understand that this provides our customers with an upfront cost, but are confident the improved design minimizes life-cycle costs moving forward.

The following tables summarize what the new class structure looks like. You will note we've borrowed ideas from the .NET designers.

Class Comparison (New vs Old)

New Classname	Old Classname	Summary
Agent (control)	Agent (control)	Functionally the same, with new features.
Manager (control)	Manager (control)	Functionally the same, with new features.
AuthoritativeEngine	n/a	Holds authoritative engine data. Presented as Manager.AuthoritativeEngine and Agent.AuthoritativeEngine.
Engine	n/a	Holds authoritative engine data received from remote authoritative engines.
EngineMap	n/a	Holds dictionary of Engine objects, indexed by IPEndPoint. Presented as Manager.EngineCache.
IPEndPoint*	n/a	Holds IPv4 and IPv6 addresses. Replaces prior use of parameters/properties for hostnames, addresses, and ports.
IPEndPointList*	n/a	Holds a list of IPEndPoint objects. Used to provide a list of local IP endpoints the socket can bind to.

Mib	SnmpMib	Now initialized with SMIv2 nodes. Improved loading of MIB files.
MibNode	Variable	Previous Variable class included node attributes as properties. Creating a new class improved clarity.
MibNodeMap	n/a	Provides MibNode lookup by OID and name. Presented as Agent.Mib.Nodes and Manager.Mib.Nodes.
MibTrap	SnmpTrap	Minor changes.
MibTrapMap	SnmpTraps	Minor changes.
Security	n/a	Holds SNMPv3 parameters and processing.
SnmpMessage	SnmpMessage	Minor changes.
n/a	SnmpTable	Table support simplified and enhanced by creating Manager.Walk() and VariableList.ToTable() methods.
n/a	SnmpTableRow	Table support simplified and enhanced by creating Manager.Walk() and VariableList.ToTable() methods.
StringList*	DartStrings	Name change for consistency.
System*	System	Will now return a list of local IPEndPoint objects.
User	n/a	Holds username, passwords, and SNMPv3 security algorithms to use. Presented as Security.User.
UserList	n/a	Holds a list of User objects. Presented as Manager.TrapUsers.
UserMap	n/a	Holds a dictionary of User objects, referenced by name. Presented as AuthoritativeEngine.Users.
Variable	SnmpVariable	No longer used to describe a MibNode, has been specialized for use in SnmpMessage.Variables and Mib.Variables.
VariableList	SnmpVariables	Updated name to reflect List nature. Presented as SnmpMessage.Variables and Mib.Variables.

Enum Comparison (New vs Old)

New Enum	**Old Enum**	**Summary**
AuthenticationConstants	n/a	Enumerates authentication hash algorithm used.
DeliveryContants*	n/a	Enumerates direction of message travel.
n/a	ErrorConstants	Obsolete enum superceded with errors defined in WinError.h
ErrorConstants	ExceptionConstants	Name updating.
GenericTrapConstants	TrapConstants	Name updating.
PduConstants	TypeConstants	Changed to perform as a bit-mask. Identifies the type of PDU.

PrivacyConstants	n/a	Enumerates encryption algorithm used.
ReportConstants	n/a	Security.ReportFlag uses this flag to indicate a Report message should be sent to the requestor.
SecuirtyLevelConstants	n/a	Enumerates user security level.
ThreadingConstants*	n/a	Enumerates threading technique to employ.
UsageConstants	n/a	Identifies what a MibNode object is used for.
VariableAccessConstants	AccessConstants	Name updating.
n/a	VariableExceptionConstants	Functionality replaced with PduConstants.
VariableStatusConstants	StatusConstants	Name updating.
VariableTypeConstants	VariableTypeConstants	Name updating.

Common Usage Scenarios

Agent Responds to Requests from Managers

Old VB6 code

```
' Start listening on port 161
Agent1.Open 161

' Request event is raised when request message arrives and is decoded
Private Sub Agent1_Request()
    ' Agent1.Message contains decoded request
End Sub

Private Sub Agent1_Response()
    ' Agent1.Message now contains the default Response
End Sub
```

New VB6 code

```
' Add all users to AuthoritativeEngine so we can respond to v3 requests
Dim user as New User
user.Name = "joe"
' Set other user properties for authentication and privacy if required
Agent1.AuthoritativeEngine.Users.Add user
' Add other users ...

' Use "Start" to listen on port 161
' Use new IPEndPoint instead of Nothing if:
'   1. IPv6 interface is desired
'   2. Selection of non-default IP interface is desired
'   3. Selection of special port is desired
' Use additional (non-default) parameter to specify FreeThreading for console
applications
Agent1.Start Nothing

' Request event is raised when request messages arrive
Private Sub Agent1_Request(ByVal requestMessage As DartSnmpCtl.ISnmpMessage)
    Dim response As SnmpMessage
    Set response = Agent1.CreateResponse(requestMessage)
    Agent1.SendResponse response, requestMessage.Origin
End Sub
```

To summarize, the new interface:

- removes the SnmpMessage.Message property in favor of making it a parameter of the Request event
- removes the Response event in favor of using CreateResponse() and SendResponse()
- provides a more declarative and explicit interface
- adds free-threading support for console, web, service applications
- adds user-based security for v3 by configuring AuthoritativeEngine properties

Agent Sends a Trap

Old VB6 code

```
' Open an ephemeral port on a new Agent for sending a trap
Agent2.Open

' Initialize Trap Message
Agent2.Message.Reset
Agent2.Message.Type = snmpTrap1
Agent2.Message.GenericTrap = snmpWarmStart
Agent2.Message.Enterprise = "MyEnterprise"
'Add sysUpTime Variable to Trap
Dim var As New SnmpVariable
var.Oid = Mib.Variables.GetOIDFromName("sysUpTime")
var.Type = snmpTimeTicks
var.value = (GetTickCount - startTime) / 10 ' hundredths of a second
Agent2.Message.Variables.Add var
' Set destination and Send
Agent2.TrapManagers.Clear
Agent2.TrapManagers.Add txtDestination.Text
Agent2.Send
```

New VB6 code

```
' No need to Open a port...SendTrap takes care of that...reuse SysUpTime from Agent1
Dim msg as New SnmpMessage
msg.Type = pduTrap1
msg.GenericTrap = trapWarmStart
msg.Enterprise = "MyEnterprise"
' Agent1.SysUpTime is used for first variable value during trap encoding
' Send trap to IPEndPoint (agentEndPoint) initialized elsewhere...uses an ephemeral
port
Agent1.SendTrap msg, agentEndPoint, Nothing
```

To summarize, the new interface:

- removes the Agent.TrapManagers property in favor of explicit use of an IPEndPoint in Agent.SendTrap (this enables IPv6 addressing)
- automatically uses Agent.SysUpTime property as required for first variable binding
- operates independently of Agent.Start method (except for initialization of SnmpMessage.SysUpTime

property)
- adds user-based security for v3 by setting SnmpMessage.Security.User properties (not shown)

Manager Receives a Trap or Inform Request

Old VB6 code

```
' Open a port for receiving traps
Manager1.Open 162

' Open a port for receiving traps...previous manager did not process inform requests
Private Sub Manager1_Trap()
  'Fires when manager receives a trap
  'Add trap info to trap log
  trapLog = trapLog + "Trap received @ " & Now & " from host " + Manager1.AgentName
+ vbCrLf
  Dim var As SnmpVariable
  For Each var In Manager1.Message.Variables
    trapLog = trapLog + var.Oid + " " + v.value + vbCrLf
  Next
End Sub
```

New VB6 code

```
' Open a port for receiving traps and inform requests...binds to IPv4 port 162 by
default
Manager1.Start Nothing

Private Sub Manager1_Trap(ByVal trapMessage As DartSnmpCtl.ISnmpMessage)
  ' Trap message has arrived. No acknowlegement required. ToString() method replaces
old code.
  trapLog = trapLog + trapMessage.ToString() + vbCrLf
End Sub

Private Sub Manager1_Inform(ByVal informMessage As DartSnmpCtl.ISnmpMessage)
  ' Inform message has arrived. Send back a response (acknowlegement)
  Dim message As SnmpMessage
  Set message = Manager1.CreateInformResponse(informMessage)
  Manager1.SendResponse message, informMessage.Origin
End Sub
```

To summarize, the new interface:

- adds support for responding to inform requests from other managers
- SnmpMessage, VariableList and Variable have new ToString() method that creates a readable description of the object

Manager Sends a Get Request

Old VB6 code

```
' Open an ephemeral port
Manager1.Open
```

```
Manager1.AgentName = agentHostName
Manager1.AgentPort = 161
Manager1.Timeout = 5000
Manager1.Message.Reset
Dim var As New SnmpVariable
'Add ".0" to the end of non-table OID
var.Oid = Manager1.Mib.Variables.GetOIDFromName("sysDescr") + ".0";
Manager1.Message.Variables.Add var
Manager1.Send
textBoxSysDescr.Text = Manager1.Variables[1].Value
Manager1.Close
```

New VB6 code

```
' No need to Open a port...GetResponse() always creates a new socket for its use
Dim request as new SnmpMessage
request.Variables.Add Manager1.Mib.Nodes(NodeNames.sysDescr).CreateVariable
request.Type = pduGet1
Dim response As SnmpMessage
' agentEndPoint describes any IPv4 or IPv6 address, "Nothing" indicates any local
IPEndPoint may be used
Set response = Manager1.GetResponse(request, agentEndPoint, Nothing)
textBoxSysDescr.Text = response.Variables[0].Value
```

To summarize, the new interface:

- removes the need to open/close a socket
- can use blocking, asynchronous, or pseudo-blocking (while processing message loop), whereas old interface was blocking or asynchronous only
- using the asynchronous option, can now make multiple requests in parallel (worker threads operate independently)
- VariableList is now indexed using a 0-based integer
- adds user-based security for v3 by setting SnmpMessage.Security.User properties (not shown)

Manager Gets a Table

Old VB6 code

```
' Open a port for getting table
Manager1.Open
Manager1.AgentPort = 161
Manager1.AgentName = agentHostname
Dim tableVariables As New SnmpVariables
Dim tableOid As String
'find table's Oid in mib
tableOid = Manager1.Mib.Variables.GetOIDFromName("ifTable")
Dim v As New SnmpVariable
v.Oid = tableOid
'start with table's Oid and send getnext requests until end of table is reached
(called a "Walk")
Do True
  Manager1.Message.Reset
  Manager1.Message.Type = snmpGetNext1
  Manager1.Message.Variables.Add v
  'Stop if error occurs
  Manager1.Send
  'Add variable in response to tableVariables collection
  'Stop if response is not part of table
  If Mid(Manager1.Message.Variables(1).Oid, 1, Len(tableOid)) = tableOid Then
    tableVariables.Add Manager1.Message.Variables(1)
  Else
    ' Id does not fall under table tree, so exit loop
    Exit Do
  End If
  'Use Oid in response for next getnext request
  v.Oid = Manager1.Message.Variables(1).Oid
Loop
' Formating code not shown because it was so complex
```

New VB6 code

```
' No need to Open a port...Walk takes care of that
' table is a 2-dimensional array
Dim variables As VariableList
Dim table() As Variant
' Walk() returns a VariableList that is built using technique indicated by specified
parameters:
'   tableOid - the "root" Oid, underwhich all MIB values will be included in the
return value
'   pdu - signals pdu version and walk 'type' (GetNext or GetBulk)
'   community - used for SNMP version 1 or 2
'   security - used for SNMP version 3
'   agentEndPoint - v4 and v6 addresses are supported
'   localEndPoint - optionally binds to the specified interface
Set variables = manager.Walk(tableOid, PduConstants.pduGetBulk2, agent.Community,
agent.security, agent.EndPoint, Nothing)
' variables holds a flat list of Variable objects in the tree
' ToTable() is used to create a 2-dimensional array that is easy to work with
(sparse tables are supported)
table = variables.ToTable(tableOid)
' How to use the table(row, col):
Dim variable As variable
For i = 0 To UBound(table, 2)
  ' Column labels found here
  Set variable = table(0, i)
  Dim name As String
  name = variable.Definition.Name ListView1.ColumnHeaders.Add , , table(0,
i).Definition.name
Next
For i = 0 To UBound(table, 1)
  Dim item As ListItem
  Set item = ListView1.ListItems.Add(, , table(i, 0).ValueAsString)
  For j = 1 To UBound(table, 2)
    item.ListSubItems.Add , , table(i, j).ValueAsString
  Next
Next
```

To summarize, the new interface provides powerful commands for retrieving and formating tables:

- a single method eliminates the need to code the Walk operation
- a single method formats a flat list of varaiables into a 2-dimensional array that is much easier to work with