# WinSNMP/MIB

## An Interface for Programming using the Management Information Base of SNMP under Microsoft® Windows™

**Version 1.0d**
**(tentative)**

**24th Oct. 1993**

**Revision history:**

| Rev | Date | Author | Author's Address |
|---|---|---|---|
| 0.1 | April 12, 1993 | (???) | |
| 0.2 | April 21, 1993 | (???) | |
| 1.0a | April 28, 1993 | (???) | |
| 1.0b | Sept. 21, 1993 | (Author: Sudhir Pendse) | sudhir@netcom.com |
| 1.0c | Sept. 30, 1993 | (Author: Sudhir Pendse) | sudhir@netcom.com |
| 1.0d | Oct. 24, 1993 | (Author: Sudhir Pendse) | sudhir@netcom.com |

# WinSNMP/MIB
# Table of Contents


1. INTRODUCTION ..... 4
   1.1 What is WinSNMP/MIB? ..... 5
2. PROGRAMMING WITH WinSNMP/MIB ..... 6
   2.1 Node ..... 6
   2.2 List Element ..... 6
   2.3 Organization of Information ..... 6
3. WinSNMP/MIB INTERFACES ..... 10
   3.1 Mapping Functions ..... 11
      3.1.1 WsmMapStrToNode() ..... 12
      3.1.2 WsmMapOidToNode() ..... 13
   3.2 Navigational Functions ..... 14
      3.2.1 WsmFindNodeChild() ..... 15
      3.2.2 WsmFindNodeParent() ..... 16
      3.2.3 WsmFindNodeNextSibling() ..... 17
      3.2.4 WsmFindNodePrevSibling() ..... 18
      3.2.5 WsmFindNextListElement() ..... 19
   3.3 Data Retrieval Functions ..... 20
      3.3.1 WsmGetNodeOid() ..... 21
      3.3.2 WsmGetNodeInt() ..... 22
      3.3.3 WsmGetNodeString() ..... 23
      3.3.4 WsmGetNodeList() ..... 24
      3.3.5 WsmGetListElementOid() ..... 25
      3.3.6 WsmGetListElementInt() ..... 26
      3.3.7 WsmGetListElementString() ..... 27
      3.3.8 WsmGetListElementList() ..... 28
   3.4 Utility Functions ..... 29
      3.4.1 WsmGetLastError() ..... 30
      3.4.2 WsmFreeList() ..... 31
      3.4.3 WsmLoadModule() ..... 32
      3.4.4 WsmUnloadModule() ..... 33
4. USAGE EXAMPLES ..... 34
   4.1 Mapping and Navigational Example ..... 34
   4.2 Module Identity Example ..... 35
   4.3 Object Identity Example ..... 36
   4.4 Object Type Example ..... 37
   4.5 Notification/Trap Type Example ..... 40
   4.6 Object Group Example ..... 41
   4.7 Module Compliance Example ..... 42
   4.8 Agent Capabilities Example ..... 45
5. DECLARATIONS ..... 49

# 1. INTRODUCTION

The WinSNMP/MIB specification is an addition to the "Windows SNMP" family of API's. The WinSNMP/MIB API is complementary to and compatible with the currently released "Windows SNMP 1.0g" API specification.

Although designed primarily for the Microsoft Windows based application software, the WinSNMP/MIB API is generic enough to be used in other operating system environments.

The WinSNMP/MIB API supports both the older SNMPv1 Concise MIB definitions, as well as the newer SNMPv2 MIB definition macros.

## 1.1 What is WinSNMP/MIB?

The WinSNMP/MIB specifications defines a set of database functions that offer management applications a method of browsing the MIB database known at the management station. The actual formation, population and implementation of the MIB database (using a MIB compiler) is left to each implementation of WinSNMP/MIB.

This version enhances the existing API (dated 28th April, 1993) , to allow for the support of the new macros defined in SNMPv2. A generic model is proposed, that would allow future changes in macros and clauses within macros to be easily supported. The existing idea of a Node is retained while the concepts of ListElements and Information Identifiers are introduced.

The WinSNMP/MIB can be viewed as an API that a management application can use to gather the information present in standard and enterprise specific MIBs, which could later be used to send and receive SNMP requests using the already published WinSNMP API.

# 2. PROGRAMMING WITH WinSNMP/MIB

## 2.1 Node

The concept of a "Node" is introduced in this document. A Node refers to any point in the hierarchical SNMP object type namespace or database or registration tree. Associated with any node in the database are several characteristics, such as OID, syntax and description. The following relationships between nodes are defined:

Node A is the PARENT of Node B if the OID of Node A matches the OID of Node B to N-1 levels, where N-1 is the total number of levels in the OID of Node A, and N is the total number of levels in the OID of Node B.

Node A is the CHILD of Node B if Node B is the PARENT of Node A, as defined in the previous rule.

Node A and B are SIBLINGS if Node A and Node B share the same PARENT node, as defined above. Sibling nodes are ordered lexicographically by the value of their Nth OID level, where N is the total number of levels in their OID.

The functions defined in this standard allow a management application to navigate the tree of nodes that makes up a MIB database, and to retrieve the characteristics of any of the nodes in that database.

The characteristics of a node are of the following data types: OBJECT IDENTIFIER, OCTETS (strings), INTEGER, and List.

## 2.2 List Element

In addition to the concept of "Node" another simple concept of a singly linked list element "ListElement" is also introduced. Like the Node, the ListElement also has information associated with it, foremost of which is the Type.

The Type of the List Element and the Node determines what additional information is associated with it.

The characteristics or information associated with a list element is of the following data type: OBJECT IDENTIFIER, OCTETS (strings), INTEGER, and List.

## 2.3 Organization of Information

The Node and ListElement structures can be viewed as "containers", that contain different types of Nodes and ListElements.

A piece of information is uniquely identified by an Information Identifier.

Each Node and List Element has one common characteristic, which is identified by the information identifier "MIB_INFO_TYPE" which gives information about the type of Node or ListElement that it "contains". Additional information available from that Node or ListElement depends on the Type of Node or ListElement.

Typically macros and assignments whose value is of type OBJECT IDENTIFIER are Node's, while list of information within them become Lists

The currently supported Node types along with the associated information for each node type is presented below.

| NODE TYPES | INFORMATION IDENTIFIER |
|---|---|
| WSM_MODULE_IDENTITY | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID(OID))<br>WSM_INFO_LAST_UPDATED (OCTETS)<br>WSM_INFO_ORGANIZATION (OCTETS)<br>WSM_INFO_CONTACT (OCTETS)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REVISIONS (List: type WSM_REV_LIST) |
| WSM_OBJECT_IDENTITY | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS) |
| WSM_OBJECT_TYPE | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_SYNTAX (INT)<br>WSM_INFO_SYNTAXSIZES (List: type WSM_SIZE_LIST)<br>WSM_INFO_SYNTAXVALS (List: type WSM_VALUE_LIST)<br>WSM_INFO_TEXTCONV_NAME (OCTETS)<br>WSM_INFO_TEXTCONV_DISPHINT (OCTETS)<br>WSM_INFO_TEXTCONV_STATUS (INT)<br>WSM_INFO_TEXTCONV_DESCR (OCTETS)<br>WSM_INFO_TEXTCONV_REFER (OCTETS)<br>WSM_INFO_UNITS (OCTETS)<br>WSM_INFO_MAXACCESS (INT)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS)<br>WSM_INFO_INDEXES (List:type WSM_INDEX_LIST)<br>WSM_INFO_AUGMENTS (OCTETS)<br>WSM_INFO_DEFVAL (OCTETS) |
| WSM_NOTIFICATION_TYPE | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_OBJECTS (List:type WSM_OBJECT_LIST)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS)<br>WSM_INFO_ENTERPRISE (OID)<br>WSM_INFO_INTVALUE (INT) |
| WSM_OBJECT_GROUP | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_OBJECTS (List: type WSM_OBJECT_LIST)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS) |

| | |
|---|---|
| WSM_MODULE_COMPLIANCE | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS)<br>WSM_INFO_MCMODULES (List: type WSM_MCMOD_LIST) |
| WSM_AGENT_CAPABILITIES | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_OID (OID)<br>WSM_INFO_PRODUCTREL (OCTETS)<br>WSM_INFO_STATUS (INT)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_REFERENCE (OCTETS)<br>WSM_INFO_ACMODULES(List: type WSM_ACMOD_LIST) |

The currently supported ListElement types along with the associated information for each List Element type is presented below.

| LIST ELEMENT TYPES | INFORMATION IDENTIFIER |
|---|---|
| WSM_REV_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_REVISION (OCTETS)<br>WSM_INFO_DESCRIPTION (OCTETS) |
| WSM_INDEX_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME(OCTETS)<br>WSM_INFO_IMPLIED (INT) |
| WSM_SIZE_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_MINSIZE (INT)<br>WSM_INFO_MAXSIZE (INT) |
| WSM_VALUE_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_VALUEINT (INT)<br>WSM_INFO_VALUELABEL (OCTETS) |
| WSM_OBJECT_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME(OCTETS) |
| WSM_MCMOD_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_MODULE (OID)<br>WSM_INFO_MANDGROUPS(List: type WSM_OBJECT_LIST)<br>WSM_INFO_MCOMPS (List: type WSM_MCCOMP_LIST) |
| WSM_MCCOMP_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_GROUPTYPE (INT) [ yes/no ]<br>WSM_INFO_OBJECTTYPE (INT)[ yes/no ]<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_DESCRIPTION (OCTETS)<br>WSM_INFO_SYNTAX (INT)<br>WSM_INFO_SYNTAXSIZES (List: type WSM_SIZE_LIST)<br>WSM_INFO_SYNTAXVALS (List: type WSM_VALUE_LIST)<br>WSM_INFO_WRITESYNTAX (INT)<br>WSM_INFO_WSYNTAXSIZES (List: type WSM_SIZE_LIST)<br>WSM_INFO_WSYNTAXVALS (List: type WSM_VALUE_LIST)<br>WSM_INFO_MINACCESS (INT) |
| WSM_ACMOD_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_SUPPORT (OID)<br>WSM_INFO_INCLUDES(List: type WSM_OBJECT_LIST)<br>WSM_INFO_VARIABLES(List: type WSM_ACVAR_LIST) |

| | |
|---|---|
| WSM_ACVAR_LIST | WSM_INFO_TYPE (INT)<br>WSM_INFO_NAME (OCTETS)<br>WSM_INFO_SYNTAX (INT)<br>WSM_INFO_SYNTAXSIZES (List: type WSM_SIZE_LIST)<br>WSM_INFO_SYNTAXVALS (List: type WSM_VALUE_LIST)<br>WSM_INFO_WRITESYNTAX (INT)<br>WSM_INFO_WSYNTAXSIZES (List: type WSM_SIZE_LIST)<br>WSM_INFO_WSYNTAXVALS (List: type WSM_VALUE_LIST)<br>WSM_INFO_ACCESS (INT)<br>WSM_INFO_CREATIONS (List: type WSM_OBJECT_LIST)<br>WSM_INFO_DEFVAL (OCTETS)<br>WSM_INFO_DESCRIPTION (OCTETS) |

## 3. WinSNMP/MIB INTERFACES

This section comprises the function reference for WinSNMP/MIB. The interface functions can be broken up into four main categories:

1. Mapping Functions:
2. Navigational Functions:
3. Data Retrieval Functions:
4. Utility Functions

Basically this API allows an application program to retrieve a Node on the registration tree and get the information associated with it. So any Macro that does not evaluate to an object identifier, and hence is not part of the registration tree, cannot be retrieved using this API.

The Version1 TRAP-TYPE macro is one such macro whose value is an integer and not an object identifier. For the purposes of this API, it is assumed that the v1 TRAP-TYPE macro, with its enterprise and integer value is mapped to a v2 NOTIFICATION-TYPE macro with a value of { enterprise.0.trapid } or the corresponding standard snmp trap. (As per the Coexistence document (RFC 1452) )

The WinSNMP/MIB API tries to use the data types defined in the WinSNMP API document, to promote consistency among the WinSNMP/* API's.

The following RFC's have been used as guidelines for the information in this document.

- RFC 1212 (Concise MIB definitions) for the SNMPv1 OBJECT-TYPE macro definitions.

- RFC1215 (Trap definitions) for the SNMPv1 TRAP-TYPE macro definitions.

- RFC 1442 (SMI for SNMPv2) for MODULE-IDENTITY, OBJECT-IDENTITY, OBJECT-TYPE and NOTIFICATION-TYPE macros.
- RFC 1443 (Textual Conventions for SNMPv2) for information on TEXTUAL-CONVENTION macro.
- RFC 1444 (Conformance Statements for SNMPv2) for information on OBJECT-GROUP, MODULE-COMPLIANCE and AGENT-CAPABILITIES macros.
- RFC 1452 (Coexistence between V1 and V2) for mapping of the v1 TRAP-TYPE macro to an object identifier
- RFC 1450 (MIB for SNMPv2) for information on standard traps/notifications.

## 3.1 Mapping Functions

The functions in this section concern the mapping of object descriptors or names and object identifiers to Node handles. These functions are a starting point for the WinSNMP/MIB implementation interaction, since as mentioned earlier, one must first get a handle to a node on the registration tree, to get information about it, or to navigate from it to another node.

The functions in this section are:

| Return Type | Procedure Name | Parameters |
|---|---|---|
| HWSM_NODE | WsmMapStrToNode() | IN LPSTR nodeString |
| HWSM_NODE | WsmMapOidToNode() | IN smiLPOID oid |

### 3.1.1 WsmMapStrToNode()

The **WsmMapStrToNode** function returns a handle to the node associated with the input string.

**Syntax**

```
HWSM_NODE          WsmMapStrToNode(
        IN LPSTR               nodeString);
```

| Parameter | Description |
|---|---|
| nodeString | The string "name" of the node in question. |

**Returns**

Returns a handle to the node in the database, which can be used as the starting point for subsequent navigation through the database. Returns NULL if the name cannot be found in the database, or if the name given cannot be unambiguously resolved to a node in the database. If NULL, use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | No such node exists in this MIB database. |
| WSM_NODE_AMBIGUOUS | The name given cannot be unambiguously resolved to one node in the MIB database. |

**Comments**

Simple textual names associated with SNMP object types may not be unique in the global MIB database. Thus, the Windows MIB implementation may not be able to unambiguously resolve textual names to a specific node in the database. When this condition is detected, this function will return NULL, with the extended error set to WSM_NODE_AMBIGUOUS. The calling application should retry the call, specifying a textual string of the form "module.object" (e.g., "RFC1213-MIB.sysDescr") to unambiguously refer to the node.

If the string parameter refers to an instance of an object type (e.g., "sysDescr.0") then the node corresponding to the object type is returned (e.g., "sysDescr").

### 3.1.2 WsmMapOidToNode()

The **WsmMapOidToNode** function returns a handle to the node associated with the object identifier.

**Syntax**

```
HWSM_NODE          WsmMapOidToNode(
        IN smiLPOID          oid);
```

| Parameter | Description |
|---|---|
| oid | The object identifier of the node in question. |

**Returns**

Returns a handle to the node in the database, which can be used as the starting point for subsequent navigation through the database. Returns NULL if the OID cannot be found in the database. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | No such node exists in this MIB database. |

**Comments**

If the OID parameter refers to an instance of an object type (e.g., "sysDescr.0") then the node corresponding to the object type is returned (e.g., "sysDescr").

## 3.2 Navigational Functions

The functions in this section allow the user to navigate through the registration tree of Nodes, or the singly linked list of ListElements. The Registration tree is like a inverted tree, and the navigational functions allow the user to traverse the tree, from the specified node. Similarly the user can also get the next elements of a singly linked list.

The functions in this section are:

| Return Type | Procedure Name | Parameters |
|---|---|---|
| HWSM_NODE | WsmFindNodeChild() | IN HWSM_NODE hNode |
| HWSM_NODE | WsmFindNodeParent() | IN HWSM_NODE hNode |
| HWSM_NODE | WsmFindNodeNextSibling() | IN HWSM_NODE hNode |
| HWSM_NODE | WsmFindNodePrevSibling() | IN HWSM_NODE hNode |
| HWSM_LISTELEMENT | WsmFindNextListElement() | IN HWSM_LISTELEMENT hListElement |

### 3.2.1 WsmFindNodeChild()

The **WsmFindNodeChild** function returns a handle to the first child node, if any exist, of the specified node.

**Syntax**

```
HWSM_NODE        WsmFindNodeChild(
        IN HWSM_NODE        hNode);
```

| Parameter | Description |
|---|---|
| hNode | The node for which the first child is desired. |

**Returns**

Returns a handle to the first child node, if any exist, of the specified node. Returns NULL if the first child cannot be found. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_NONE | The specified node has no children in this MIB database. |

**Comments**

None.

### 3.2.2 WsmFindNodeParent()

The **WsmFindNodeParent** function returns a handle to the parent node, if one exists, of the specified node.

**Syntax**

```
HWSM_NODE          WsmFindNodeParent(
        IN HWSM_NODE       hNode);
```

| Parameter | Description |
|---|---|
| hNode | The node for which the parent node is desired. |

**Returns**

Returns a handle to the parent node, if it exists, of the specified node. Returns NULL if the parent node cannot be found. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_NONE | The specified node has no parent in this MIB database. |

**Comments**

A node with no parent should be considered the global root of the MIB database on this machine. Only one such node should exist per database.

### 3.2.3 WsmFindNodeNextSibling()

The **WsmFindNodeNextSibling** function returns a handle to the next sibling node, if one exists, of the specified node.

**Syntax**

```
HWSM_NODE        WsmFindNodeNextSibling(
        IN HWSM_NODE        hNode);
```

| Parameter | Description |
|---|---|
| hNode | The node for which the next sibling node is desired. |

**Returns**

Returns a handle to the next sibling node, if one exists, of the specified node. Returns NULL if there are no more siblings of the specified node. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_NONE | The specified node has no lexicographically next sibling in this MIB database. |

**Comments**

Sibling nodes are ordered lexicographically by the value of their Nth OID level, where N is the total number of levels in their OID.

### 3.2.4 WsmFindNodePrevSibling()

The **WsmFindNodePrevSibling** function returns a handle to the previous sibling node, if one exists, of the specified node.

**Syntax**

```
HWSM_NODE        WsmFindNodePrevSibling(
        IN HWSM_NODE        hNode);
```

| Parameter | Description |
|---|---|
| hNode | The node for which the previous sibling node is desired. |

**Return**

Returns a handle to the previous sibling node, if one exists, of the specified node. Returns NULL if there are no previous siblings of the specified node. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_NONE | The specified node has no lexicographically previous sibling in this MIB database. |

**Comments**

Sibling nodes are ordered lexicographically by the value of their Nth OID level, where N is the total number of levels in their OID.

### 3.2.5 WsmFindNextListElement()

The **WsmFindNextListElement** function returns a handle to the next list element, if one exists, of the specified list element.

**Syntax**

```
HWSM_LISTELEMENT        WsmFindNextListElement(
    IN HWSM_LISTELEMENT hListElement);
```

| Parameter | Description |
|---|---|
| hListElement | The list element for which the next list element is desired. |

**Returns**

Returns a handle to the next list element, if it exists, of the specified list element. Returns NULL if there no additional elements in the list. Use WsmGetLastError to obtain extended error information.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_NONE | The specified list element has no successive element in the list. |

**Comments**

Unlike the Node, the ListElements are organized in a singly linked list, with no mechanism to go backwards.

## 3.3 Data Retrieval Functions

The functions in this section allow the user to retrieve the data associated with a Node or List Element.

The data associated with a Node or ListElement is of the following types: Integer, Octets (String), Object Identifier and List.

The data is identified by an Information Identifier, which are #defined integer constants.

The functions in this section are:

| Return Type | Procedure Name | Parameters |
|---|---|---|
| SNMPAPI_STATUS | WsmGetNodeOid() | IN HWSM_NODE hNode,<br>IN smiINT infoID,<br>OUT smiLPOID oid |
| SNMPAPI_STATUS | WsmGetNodeInt() | IN HWSM_NODE hNode,<br>IN smiINT infoID,<br>OUT smiLPINT intval |
| SNMPAPI_STATUS | WsmGetNodeString() | IN HWSM_NODE hNode,<br>IN smiINT infoID,<br>IN OUT smiLPINT strLength,<br>OUT LPSTR strbuf |
| SNMPAPI_STATUS | WsmGetNodeList() | IN HWSM_NODE hNode,<br>IN smiINT infoID,<br>OUT LPHWSM_LISTELEMENT hListElem |
| SNMPAPI_STATUS | WsmGetListElementOid() | IN HWSM_LISTELEMENT hListElem,<br>IN smiINT infoID,<br>OUT smiLPOID oid |
| SNMPAPI_STATUS | WsmGetListElementInt() | IN HWSM_LISTELEMENT hListElem,<br>IN smiINT infoID,<br>OUT smiLPINT intval |
| SNMPAPI_STATUS | WsmGetListElementString() | IN HWSM_LISTELEMENT hListElem,<br>IN smiINT infoID,<br>IN OUT smiLPINT strLength,<br>OUT LPSTR strbuf |
| SNMPAPI_STATUS | WsmGetListElementList() | IN HWSM_LISTELEMENT hListElem,<br>IN smiINT infoID,<br>OUT LPHWSM_LISTELEMENT hListE |

### 3.3.1 WsmGetNodeOid()

The **WsmGetNodeOid** function returns the MIB information for the specified node, that is of type object identifier, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetNodeOid(
    IN HWSM_NODE    hNode,
    IN smiINT       infoID,
    OUT smiLPOID    lpOid)
```

| Parameter | Description |
|---|---|
| hNode | The object type (node) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpOid | Receives the OID value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | The specified node does not exist in this MIB database. |
| WSM_NODE_NOINFO | No information is available for the specified information identifier and node. |

**Comments**

It is the responsibility of the application making this call, to free the memory associated with the filled object identifier structure.

### 3.3.2 WsmGetNodeInt()

The **WsmGetNodeInt** function returns the MIB information for the specified node, that is of type Integer, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetNodeInt(
    IN HWSM_NODE    hNode,
    IN smiINT       infoID,
    OUT smiLPINT    lpInt)
```

| Parameter | Description |
|---|---|
| hNode | The object type (node) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpInt | Receives the integer value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | The specified node does not exist in this MIB database. |
| WSM_NODE_NOINFO | No information is available for the specified information identifier and node. |

**Comments**

None

### 3.3.3 WsmGetNodeString()

The **WsmGetNodeString** function returns the MIB information for the specified node, that is of type string, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetNodeString(
    IN HWSM_NODE        hNode,
    IN smiINT           infoID,
    IN OUT smiLPINT     lpMaxStrLen,
    OUT LPSTR           lpStr)
```

| Parameter | Description |
|---|---|
| hNode | The object type (node) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpMaxStrLen | Specifies the buffer size pointed to by lpStr. Upon return, it contains the size of the buffer used. |
| lpStr | Receives the string value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | The specified node does not exist in this MIB database. |
| WSM_NODE_NOINFO | No information is available for the specified information identifier and node. |
| WSM_NODE_TRUNCATED | The string returned was truncated. |

**Comments**

None.

### 3.3.4 WsmGetNodeList()

The **WsmGetNodeList** function returns the MIB information for the specified node, that is of type List, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetNodeList(
    IN HWSM_NODE                hNode,
    IN smiINT                   infoID,
    OUT LPHWSM_LISTELEMENT      lpHListElement)
```

| Parameter | Description |
|---|---|
| hNode | The object type (node) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpHListElement | Receives the handle to the List Element that is at the start of the List, for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_NODE_UNKNOWN | The specified node does not exist in this MIB database. |
| WSM_NODE_NOINFO | No information is available for the specified information identifier and node. |

**Comments**

It is the responsibility of the application making this call, to free the memory associated with the singly linked list structure using the WsmFreeList() call.

### 3.3.5 WsmGetListElementOid()

The **WsmGetListElementOid** function returns the MIB information for the specified list element, that is of type object identifier, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetListElementOid(
    IN HWSM_LISTELEMENT         hListElement,
    IN smiINT                   infoID,
    OUT smiLPOID                lpOid)
```

| Parameter | Description |
|---|---|
| hListElement | The object type (list element) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpOid | Receives the OID value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_UNKNOWN | The specified list element does not exist in this MIB database. |
| WSM_LISTELEMENT_NOINFO | No information is available for the specified information identifier and list element. |

**Comments**

It is the responsibility of the application making this call, to free the memory associated with the filled object identifier structure.

### 3.3.6 WsmGetListElementInt()

The **WsmGetListElementInt** function returns the MIB information for the specified list element, that is of type Integer, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS              WsmGetListElementInt(
    IN HWSM_LISTELEMENT     hListElement,
    IN smiINT               infoID,
    OUT smiLPINT            lpInt)
```

| Parameter | Description |
|---|---|
| hListElement | The object type (list element) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpInt | Receives the integer value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_UNKNOWN | The specified list element does not exist in this MIB database. |
| WSM_LISTELEMENT_NOINFO | No information is available for the specified information identifier and list element. |

**Comments**

None

### 3.3.7 WsmGetListElementString()

The **WsmGetListElementString** function returns the MIB information for the specified list element, that is of type string, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetListElementString(
    IN HWSM_LISTELEMENT         hListElement,
    IN smiINT                   infoID,
    IN OUT smiLPINT             lpMaxStrLen,
    OUT LPSTR                   lpStr)
```

| Parameter | Description |
|---|---|
| hListElement | The object type (list element) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpMaxStrLen | Specifies the buffer size pointed to by lpStr. Upon return, it contains the size of the buffer used. |
| lpStr | Receives the string value for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_UNKNOWN | The specified list element does not exist in this MIB database. |
| WSM_LISTELEMENT_NOINFO | No information is available for the specified information identifier and list element. |
| WSM_LISTELEMENT_TRUNCATED | The string returned was truncated. |

**Comments**

None.

### 3.3.8 WsmGetListElementList()

The **WsmGetListElementList** function returns the MIB information for the specified list element, that is of type List, and identified by the infoID.

**Syntax**

```
SNMPAPI_STATUS                  WsmGetListElementList(
    IN HWSM_LISTELEMENT         hListElement,
    IN smiINT                   infoID,
    OUT LPHWSM_LISTELEMENT      lpHListElement)
```

| Parameter | Description |
|---|---|
| hListElement | The object type (list element) for which data is desired. |
| infoID | The information identifier. See Section 2.3 for possible values. |
| lpHListElement | Receives the handle to the List Element that is at the start of the List, for the specified information identifier. |

**Returns**

Returns SNMPAPI_SUCCESS if the data is successfully retrieved. Use WsmGetLastError to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_UNKNOWN | The specified list element does not exist in this MIB database. |
| WSM_LISTELEMENT_NOINFO | No information is available for the specified information identifier and list element. |

**Comments**

It is the responsibility of the application making this call, to free the memory associated with the singly linked list structure, using the WsmFreeList() call.

## 3.4 Utility Functions

The functions in this section, in reality, fall in the "miscellaneous" category, or general purpose utility functions.

The functions in this section are:

| Return Type | Procedure Name | Parameters |
|---|---|---|
| SNMPAPI_STATUS | WsmGetLastError() | None |
| SNMPAPI_STATUS | WsmFreeList() | IN HWSM_LISTELEMENT hListElem |
| SNMPAPI_STATUS | WsmLoadModule() | IN LPSTR modName |
| SNMPAPI_STATUS | WsmUnloadModule() | IN LPSTR modName |

### 3.4.1 WsmGetLastError()

The **WsmGetLastError** function returns the reason why the last operation failed.

**Syntax**

SNMPAPI_STATUS WsmGetLastError(void);

**Parameters**

None

**Return**

This function returns the last WinSNMP/MIB error that occurred. This function should be called immediately after any API call that fails, as the value is overwritten after each API call.

### 3.4.2 WsmFreeList()

The **WsmFreeList** function frees the singly linked list structure allocated by the WinSNMP/MIB implementation.

**Syntax**

```
SNMPAPI_STATUS                  WsmFreeList(
        IN HWSM_LISTELEMENT     hListElement)
```

| Parameter | Description |
|---|---|
| hListElement | The **start** of the singly linked list, that is to be made free. |

**Returns**

Returns SNMPAPI_SUCCESS if the list is successfully made free. Use WsmGetLastError() to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_LISTELEMENT_UNKNOWN | The specified list element does not exist in this MIB database. |

**Comments**

None.

### 3.4.3 WsmLoadModule()

The **WsmLoadModule** function loads information from a MIB module into the winSNMP/MIB implementation database.

**Syntax**

```
SNMPAPI_STATUS      WsmLoadModule(
        IN LPSTR    module);
```

| Parameter | Description |
|---|---|
| module | Identifies the MIB module to be loaded. |

**Return**

Returns SNMPAPI_SUCCESS if the specified module is successfully loaded. Use WsmGetLastError() to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_MODULE_NOTLOADED | Indicates that the module parameter is unknown. |

**Comments**

The module parameter identifies the ASN.1 descriptive name of the MIB module, e.g., "FIZBIN-MIB" taken from "FIZBIN-MIB DEFINITION ::= BEGIN".

If the information is already loaded, it might choose to increment a "no_of_users" flag. [ Comments are requested on this issue. Is it mandatory for applications to first load a module before it retrieves information from it, or can it assume that some modules are always loaded eg: MIBII ]

### 3.4.4 WsmUnloadModule()

The **WsmUnloadModule** function unloads information about a MIB module, from the winSNMP/MIB implementation database.

**Syntax**

```
SNMPAPI_STATUS      WsmUnloadModule(
        IN LPSTR    module);
```

| Parameter | Description |
|---|---|
| module | Identifies the MIB module to be loaded. |

**Return**

Returns SNMPAPI_SUCCESS if the specified module is successfully unloaded. Use WsmGetLastError() to obtain extended error information, in case the function returns SNMPAPI_FAILURE.

| WsmGetLastError() | Description |
|---|---|
| WSM_MODULE_NOTLOADED | Indicates that the module parameter is unknown. |
| WSM_MODULE_NOTUNLOADED | Indicates that the module parameter could not be unloaded. |

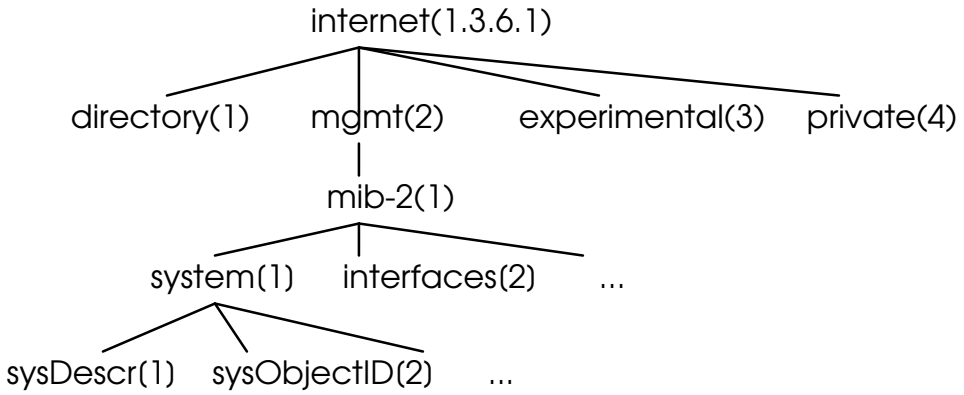
**Comments**

The module parameter identifies the ASN.1 descriptive name of the MIB module, e.g., "FIZBIN-MIB" taken from "FIZBIN-MIB DEFINITION ::= BEGIN".

It might choose to decrement the "no_of_users" flag and remove it from memory when it becomes zero. [ Comments are requested on this issue. Is it mandatory for applications to first load a module before it retrieves information from it, or can it assume that some modules are always loaded eg: MIBII ]

# 4. USAGE EXAMPLES

## 4.1 Mapping and Navigational Example




Above is an example registration tree:

Lets say we want to get a handle to the sysDescr node.

We could use
```
hNode = WsmMapStrToNode("sysDescr");    OR
hNode = WsmMapOidToNode(&oid);
```
where oid is a object identifier structure filled in with the equivalent of 1.3.6.1.2.1.1.1

One could also have got a handle to the Node "internet"
```
hNode = WsmMapStrToNode("internet");
```
and then used the Navigational functions like:
```
hNodeDirectory = WsmFindNodeChild(hNode);
```
which will give the handle to the first child node (directory). Now one can get all its siblings.
```
hNodeMgmt = WsmFindNextSibling(hNodeDirectory);
```
will return a handle to the mgmt node.
```
hNodeMib2 = WsmFindNodeChild(hNodeMgmt);
```
will return a handle to the mib-2 node.
```
hNodeSystem = WsmFindNodeChild(hNodeMib2);
```
will return a handle to the system group.
```
hNodeSysDescr = WsmFindNodeChild(hNodeSystem);
```
will return a handle to the system group.

As a side note:
```
hNode = WsmFindNodeParent(hNodeSysDescr);
```
will return a handle to the system group node. The call
```
hNode = WsmFindPrevSibling(hNodeMgmt);
```
will return a handle to the directory node. And the call
```
hNode = WsmFindNodeChild(hNodeSysDescr);
```
will return a NULL handle, since it has no children.

```
hListElement2 = WsmFindNextListElement(hListElement1)
```
will return the next element in the linked list of list elements.

## 4.2 Module Identity Example

```
abcModuleIdentity   MODULE-IDENTITY
        LAST-UPDATED        "9210070433Z"
        ORGANIZATION        "Org Name"
        CONTACT-INFO        "Contact Name"
        DESCRIPTION         "Description Text"
        REVISION            "9210070433Z"
        DESCRIPTION         "Description about revision"
        ::= { pqr 1 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_MODULE_IDENTITY in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcModuleIdentity" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of pqr.1

WsmGetNodeString(hNode, WSM_INFO_LAST_UPDATED, &strLength, buf);
will return "9210070433Z" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_ORGANIZATION, &strLength, buf);
will return "Org Name" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_CONTACT, &strLength, buf);
will return "Contact Name" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeList(hNode, WSM_INFO_REVISIONS, &hListElement);
will return a handle to the head of a  list made up of ListElements of type WSM_REV_LIST.
For each list element in the list (in the above case only one element in the list)

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_REV_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_REVISION, &strLength, buf);
will return the Revision information in the buffer. ("9210070433Z")

WsmGetListElementString(hListElement, WSM_INFO_DESCRIPTION, &strLength, buf);
will return the associated description information. ("Description about revision")

## 4.3 Object Identity Example

```
abcObjectIdentity   OBJECT-IDENTITY
        STATUS              current
        DESCRIPTION         "Description Text"
        ::= { pqr 2 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_IDENTITY in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcObjectIdentity" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of pqr.2

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return WSM_STATUS_CURRENT in tmpint.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

## 4.4 Object Type Example

```
abcObjectType   OBJECT-TYPE
        SYNTAX          INTEGER {
                        choicelabel1 (1),
                        choicelabel2 (2)
                        }
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION "Description Text"
        ::= { pqr 3 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_TYPE in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcObjectType" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of pqr.3

WsmGetNodeInt(hNode, WSM_INFO_SYNTAX, &tmpint);
will return SNMP_SYNTAX_INT in tmpint. In case of the use of Textual conventions, it will return the basic SNMP data type that the Textual convention maps to.

WsmGetNodeList(hNode, WSM_INFO_SYNTAXSIZES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_SIZE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_SIZE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_MINSIZE, &tmpint);
will return the minimum size.

WsmGetListElementInt(hListElement, WSM_INFO_MAXSIZE, &tmpint);
will return the maximum size.

In the case of SIZE( 1 | 3 | 10..12), there would be three list elements with the first element having min = 1, max = 1; second element having min = 3, max = 3; and the third having min = 10, and max = 12. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeList(hNode, WSM_INFO_SYNTAXVALS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_VALUE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_VALUE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_VALUEINT, &tmpint);
will return the integer value.

WsmGetListElementString(hListElement, WSM_INFO_VALUELABEL, &strLength, buf);
will return the associated label.

In the above case, there would be two list elements with the first element having valueint = 1, valuelabel = "choicelabel1"; and the second element having valueint = 2, valuelabel = "choicelabel2".

WsmGetNodeString(hNode, WSM_INFO_TEXTCONV_NAME, &strLength, buf);
will return the name of the textual convention used. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_TEXTCONV_DISPHINT, &strLength, buf);
will return the display hint information associated with the textual convention used. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeInt(hNode, WSM_INFO_TEXTCONV_STATUS, &tmpint);
will return the status of the textual convention used. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_TEXTCONV_DESCR, &strLength, buf);
will return the description information associated with the textual convention used. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_TEXTCONV_REFER, &strLength, buf);
will return the reference information associated with the textual convention used. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_UNITS, &strLength, buf);
will return the UNITS information. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeInt(hNode, WSM_INFO_MAXACCESS, &tmpint);
will return WSM_ACCESS_READONLY in tmpint.

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return WSM_STATUS_MANDATORY in tmpint.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

WsmGetNodeList(hNode, WSM_INFO_INDEXES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_INDEX_LIST. For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_INDEX_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_NAME, &tmpint);
will return the object descriptor of the columns that make up the index.

WsmGetListElementInt(hListElement, WSM_INFO_IMPLIED, &tmpint);
will return WSM_YES_FLAG or WSM_NO_FLAG if the corresponding index column uses the IMPLIED rules for instance creation.

In the case of "INDEX a, b, IMPLIED c" there would be three list elements with the first element having name = "a", implied = NO; second element having name = "b", implied = NO; and the third having name = "c", implied = YES. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_AUGMENTS, &strLength, buf);
will return the object descriptor of the table entry that this entry appends to. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_DEFVAL, &strLength, buf);
will return the default value of the attribute. Currently it is just returned as a string, but it could be returned as the appropriate datatype, based on the syntax of the attribute. That would require an additional function call that returns a filled smiVALUE structure. Since this is of more interest to agents than managers, it is ignored for now.

## 4.5 Notification/Trap Type Example

```
abcTrapType   TRAP-TYPE
        ENTERPRISE          pqr
        VARIABLES           { abcObjectType }
        DESCRIPTION         "Description Text"
        ::= 4
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_NOTIFICATION_TYPE in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcTrapType" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of { pqr.0.4 }, as per the rules specified in RFC 1452. In case of standard traps, the oid returned will be that of the standard traps defined in RFC 1450. For SNMPv2 notifications, it will return the value of the macro, which is a object identifier.

WsmGetNodeList(hNode, WSM_INFO_OBJECTS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_OBJECT_LIST. For each list element in the list (in the above case only one element in the list)

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);
will return the object descriptor of the attribute to be included in the varbind section of the trap/notification.("abcObjectType")

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return the status of the macro definition. In the above case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

WsmGetNodeString(hNode, WSM_INFO_ENTERPRISE, &strLength, buf);
will return the enterprise clause of SNMPv1 TRAP-TYPE declaration. In the above case it will return "pqr" in the buffer. For SNMPv2 NOTIFICATION-TYPE declarations, it would return WSM_NODE_NOINFO.

WsmGetNodeInt(hNode, WSM_INFO_INTVALUE, &tmpint);
will return the integer value of the SNMPv1 TRAP-TYPE declaration. In the above case it will return 4 in tmpint. For SNMPv2 NOTIFICATION-TYPE declarations, it would return WSM_NODE_NOINFO.

## 4.6 Object Group Example

```
abcGroup   OBJECT-GROUP
        OBJECTS             { abcObjectType }
        STATUS              current
        DESCRIPTION         "Description Text"
        ::= { pqr 5 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_GROUP in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcGroup" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of { pqr.5 }.

WsmGetNodeList(hNode, WSM_INFO_OBJECTS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_OBJECT_LIST.
For each list element in the list (in the above case only one element in the list)

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);
will return the object descriptor of the attributes that are part of this group.
("abcObjectType")

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return WSM_STATUS_CURRENT in tmpint.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

## 4.7 Module Compliance Example

```
abcModComp   MODULE-COMPLIANCE
        STATUS              current
        DESCRIPTION         "Description Text"
        MODULE              abcModuleIdentity
                MANDATORY-GROUPS        { abcObjectGroup }
        MODULE              -- this module
                MANDATORY-GROUPS        { groupA, groupB }
                GROUP       groupC
                DESCRIPTION "Conditions for groupC usage"
        ::= { pqr 6 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_MODULE_COMPLIANCE in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcModComp" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of { pqr.6 }.

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return WSM_STATUS_CURRENT in tmpint.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

WsmGetNodeList(hNode, WSM_INFO_MCMODULES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_MCMOD_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_MCMOD_LIST in tmpint.

WsmGetListElementOid(hListElement, WSM_INFO_MODULE, &oid);
will return a filled oid structure containing the object identifier of the module.
(In this case there would be two elements in the WSM_MCMOD_LIST. The first one will return the oid of abcModuleIdentity which is { pqr 1 } and the second element will return the oid of the module in which the module compliance macro is defined )

WsmGetListElementList(hNode, WSM_INFO_MANDGROUPS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_OBJECT_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);

will return the name of the groups. ("abcObjectGroup" in one case and "groupA" and "groupB" in the other)

WsmGetListElementList(hNode, WSM_INFO_MCOMPS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_MCCOMP_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_MCCOMP_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_GROUPTYPE, &tmpint);
will return WSM_YES_FLAG, if the CompliancePart is of type GROUP in tmpint, or return WSM_NO_FLAG if the Compliance Part is of type OBJECT. (in the above case it will return WSM_YES_FLAG)

WsmGetListElementInt(hListElement, WSM_INFO_OBJECTTYPE, &tmpint);
will return WSM_YES_FLAG, if the CompliancePart is of type OBJECT in tmpint, or return WSM_NO_FLAG if the Compliance Part is of type OBJECT.
(in the above case it will return WSM_NO_FLAG)

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);
will return the name of the group or object. ("groupC" in the above case )

WsmGetListElementString(hListElement, WSM_INFO_DESCRIPTION,
&strLength, buf);
will return the description clause associated with the group or object.
("Conditions for groupC usage" in the above case )

WsmGetListElementInt(hListElement, WSM_INFO_SYNTAX, &tmpint);
will return the syntax associated with the Compliance object.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_SYNTAXSIZES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_SIZE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_SIZE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_MINSIZE, &tmpint);
will return the minimum size.

WsmGetListElementInt(hListElement, WSM_INFO_MAXSIZE, &tmpint);
will return the maximum size.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_SYNTAXVALS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_VALUE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_VALUE_LIST in tmpint.

WsmGetListElementInt(hListElement,WSM_INFO_VALUEINT,&tmpint);
will return the integer value.

WsmGetListElementString(hListElement, WSM_INFO_VALUELABEL,
&strLength, buf);
will return the associated label.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetListElementInt(hListElement, WSM_INFO_WRITESYNTAX, &tmpint);
will return the write syntax associated with the Compliance object.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_WSYNTAXSIZES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_SIZE_LIST.
For each list element in the list
WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_SIZE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_MINSIZE, &tmpint);
will return the minimum size.

WsmGetListElementInt(hListElement, WSM_INFO_MAXSIZE, &tmpint);
will return the maximum size.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_WSYNTAXVALS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_VALUE_LIST.
For each list element in the list
WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_VALUE_LIST in tmpint.

WsmGetListElementInt(hListElement,WSM_INFO_VALUEINT,&tmpint);
will return the integer value.

WsmGetListElementString(hListElement, WSM_INFO_VALUELABEL,
&strLength, buf);
will return the associated label.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetListElementInt(hListElement, WSM_INFO_MINACCESS, &tmpint);
will return the minimum access required for the Compliance object.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

## 4.8 Agent Capabilities Example

```
abcAgentCap   AGENT-CAPABILITIES
        PRODUCT-RELEASE   "Product release number"
        STATUS            current
        DESCRIPTION       "Description Text"
        SUPPORTS          abcModuleIdentity
                INCLUDES          { abcObjectGroup }

                VARIATION     abcObjectType
                SYNTAX        INTEGER { choicelabel1(1) }
                DESCRIPTION "No choicelabel2"

        SUPPORTS          yetAnotherMibModule
                INCLUDES          { groupA, groupB }

                VARIATION     tableEntry
                CREATION-REQUIRES       { tableColumn1, tableColumn2 }
                DESCRIPTION "some text"
        ::= { pqr 7 }
```

If hNode is a handle to the above Node, then :

WsmGetNodeInt(hNode, WSM_INFO_TYPE, &tmpint);
will return WSM_AGENT_CAPABILITIES in tmpint.

WsmGetNodeString(hNode, WSM_INFO_NAME, &strLength, buf);
will return "abcAgentCap" in the buffer.

WsmGetNodeOid(hNode, WSM_INFO_OID, &oid);
will return a filled oid structure containing the equivalent of { pqr.7 }.

WsmGetNodeString(hNode, WSM_INFO_PRODUCTREL, &strLength, buf);
will return "Product Release Number" in the buffer.

WsmGetNodeInt(hNode, WSM_INFO_STATUS, &tmpint);
will return WSM_STATUS_CURRENT in tmpint.

WsmGetNodeString(hNode, WSM_INFO_DESCRIPTION, &strLength, buf);
will return "Description Text" in the buffer.

WsmGetNodeString(hNode, WSM_INFO_REFERENCE, &strLength, buf);
will return the information associated with the REFERENCE clause of the macro. But in this case it will return WSM_NODE_NOINFO.

WsmGetNodeList(hNode, WSM_INFO_ACMODULES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_ACMOD_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_ACMOD_LIST in tmpint.

WsmGetListElementOid(hListElement, WSM_INFO_SUPPORT, &oid);
will return a filled oid structure containing the object identifier of the module specified in the SUPPORTS clause.

(In this case there would be two elements in the WSM_ACMOD_LIST. The first one will return the oid of abcModuleIdentity which is { pqr 1 } and the second element will return the oid of the module "yetAnotherMibModule" )

WsmGetListElementList(hNode, WSM_INFO_INCLUDES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_OBJECT_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);
will return the name of the groups. ("abcObjectGroup" in one case and "groupA" and "groupB" in the other)

WsmGetListElementList(hNode, WSM_INFO_VARIABLES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_ACVAR_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_ACVAR_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME, &strLength, buf);
will return the name of the VARIATION clause. ("abcObjectType" and "tableEntry" in the above case )

WsmGetListElementString(hListElement, WSM_INFO_DESCRIPTION,
&strLength, buf);
will return the description clause associated with the variation.
("No choicelabel2" and "some text" in the above case )

WsmGetListElementInt(hListElement, WSM_INFO_SYNTAX, &tmpint);
will return the syntax associated with the Variation object.
(in the above case it will return SNMP_SYNTAX_INTEGER and WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_SYNTAXSIZES, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_SIZE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_SIZE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_MINSIZE, &tmpint);
will return the minimum size.

WsmGetListElementInt(hListElement, WSM_INFO_MAXSIZE, &tmpint);
will return the maximum size.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_SYNTAXVALS, &hListElement);
will return a handle to the head of a list made up of ListElements of type WSM_VALUE_LIST.
For each list element in the list

WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_VALUE_LIST in tmpint.

WsmGetListElementInt(hListElement,WSM_INFO_VALUEINT,&tmpint);
will return the integer value.

WsmGetListElementString(hListElement, WSM_INFO_VALUELABEL,
&strLength, buf);
will return the associated label.
(in the above case it will return one element in the list :1 and "choicelabel1"
and WSM_LISTELEMENT_NOINFO for the other)

WsmGetListElementInt(hListElement, WSM_INFO_WRITESYNTAX, &tmpint);
will return the write syntax associated with the Variation object.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_WSYNTAXSIZES, &hListElement);
will return a handle to the head of a list made up of ListElements of type
WSM_SIZE_LIST.
For each list element in the list
WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_SIZE_LIST in tmpint.

WsmGetListElementInt(hListElement, WSM_INFO_MINSIZE, &tmpint);
will return the minimum size.

WsmGetListElementInt(hListElement, WSM_INFO_MAXSIZE, &tmpint);
will return the maximum size.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetNodeList(hNode, WSM_INFO_WSYNTAXVALS, &hListElement);
will return a handle to the head of a list made up of ListElements of type
WSM_VALUE_LIST.
For each list element in the list
WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_VALUE_LIST in tmpint.

WsmGetListElementInt(hListElement,WSM_INFO_VALUEINT,&tmpint);
will return the integer value.

WsmGetListElementString(hListElement, WSM_INFO_VALUELABEL,
&strLength, buf);
will return the associated label.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetListElementInt(hListElement, WSM_INFO_ACCESS, &tmpint);
will return the minimum access required for the Variation object.
(in the above case it will return WSM_LISTELEMENT_NOINFO)

WsmGetListElementList(hNode, WSM_INFO_CREATIONS, &hListElement);
will return a handle to the head of a list made up of ListElements of type
WSM_OBJECT_LIST.
For each list element in the list
WsmGetListElementInt(hListElement, WSM_INFO_TYPE, &tmpint);
will return WSM_OBJECT_LIST in tmpint.

WsmGetListElementString(hListElement, WSM_INFO_NAME,

&strLength, buf);

will return the names of the attributes that are necessary for creation. (in the above case it will return WSM_LISTELEMENT_NOINFO and a list with two elements: "tableColumn1" and "tableColumn2".)

WsmGetNodeString(hNode, WSM_INFO_DEFVAL, &strLength, buf);
will return the default value of the attribute. Currently it is just returned as a string. (in the above case it will return WSM_LISTELEMENT_NOINFO)

## 5. DECLARATIONS

This section offers a prototype of the "WSNMPMIB.H" file containing common declarations for defines, typedefs and function prototypes.

The file includes "winsnmp.h", so that it can inherit and use the the same typedefs and defines for compatibility with the winSNMP API. Thus applications can retrieve the Object Identifier associated with an attribute using the WSMGetNodeOid() call, and pass the filled in data structure to the SnmpSetVb() call. Similarly the values of the WSM_INFO_SYNTAX information identifier would be compatible with the values of syntax field expected in the winSNMP API.

```
/*****************************************************************************
 * FILE:              wsnmpmib.h
 * DESCRIPTION: This file contains the defines, data structures, typedefs
 *                    and function prototypes necessary to implement the
 *                    winSNMP/MIB API
 * MODIFICATION: (10/18/93) Created
 *****************************************************************************/

#ifndef _INC_WSNMPMIB
#define _INC_WSNMPMIB

#include <windows.h>
#include <winsnmp.h>          /* includes winsnmp.h for compatibility */

/* windows related types */
typedef HANDLE HWSM_NODE,              FAR *LPHWSM_NODE;
typedef HANDLE HWSM_LISTELEMENT,       FAR *LPHWSM_LISTELEMENT;

/* status values */
#define WSM_STATUS_MANDATORY                1
#define WSM_STATUS_OPTIONAL                 2
#define WSM_STATUS_OBSOLETE                 3
#define WSM_STATUS_DEPRECATED               4
#define WSM_STATUS_CURRENT                  5

/* access values */
#define WSM_ACCESS_READONLY                 1
#define WSM_ACCESS_READWRITE                2
#define WSM_ACCESS_WRITEONLY                3
#define WSM_ACCESS_NOTACCESSIBLE            4
#define WSM_ACCESS_READCREATE               5

/* flag values */
#define WSM_YES_FLAG                        1
#define WSM_NO_FLAG                         2

/* error codes returned by WsmGetLastError(), so as not to    */
/* conflict with the error codes from winSNMP                 */
#define WSM_NODE_UNKNOWN                    200
#define WSM_NODE_AMBIGUOUS                  201
#define WSM_NODE_NONE                       202
#define WSM_NODE_NOINFO                     203
#define WSM_NODE_TRUNCATED                  204
#define WSM_LISTELEMENT_UNKNOWN             205
#define WSM_LISTELEMENT_AMBIGUOUS           206
#define WSM_LISTELEMENT_NONE                207
#define WSM_LISTELEMENT_NOINFO              208
#define WSM_LISTELEMENT_TRUNCATED           209
#define WSM_MODULE_NOTLOADED                210
#define WSM_MODULE_NOTUNLOADED              211

/* information identifiers */
#define WSM_INFO_TYPE                       1
#define WSM_INFO_NAME                       2
#define WSM_INFO_OID                        3
#define WSM_INFO_LAST_UPDATED               4
```

```
#define WSM_INFO_ORGANIZATION           5
#define WSM_INFO_CONTACT                6
#define WSM_INFO_DESCRIPTION            7
#define WSM_INFO_REVISIONS              8
#define WSM_INFO_STATUS                 9
#define WSM_INFO_REFERENCE              10
#define WSM_INFO_SYNTAX                 11
#define WSM_INFO_SYNTAXSIZES            12
#define WSM_INFO_SYNTAXVALS             13
#define WSM_INFO_TEXTCONV_NAME          14
#define WSM_INFO_TEXTCONV_DISPHINT      15
#define WSM_INFO_TEXTCONV_STATUS        16
#define WSM_INFO_TEXTCONV_DESCR         17
#define WSM_INFO_TEXTCONV_REFER         18
#define WSM_INFO_UNITS                  19
#define WSM_INFO_MAXACCESS              20
#define WSM_INFO_INDEXES                21
#define WSM_INFO_AUGMENTS               22
#define WSM_INFO_DEFVAL                 23
#define WSM_INFO_OBJECTS                24
#define WSM_INFO_ENTERPRISE             25
#define WSM_INFO_INTVALUE               26
#define WSM_INFO_OBJECTS                27
#define WSM_INFO_MCMODULES              28
#define WSM_INFO_PRODUCTREL             29
#define WSM_INFO_ACMODULES              30
#define WSM_INFO_REVISION               31
#define WSM_INFO_IMPLIED                32
#define WSM_INFO_MINSIZE                33
#define WSM_INFO_MAXSIZE                34
#define WSM_INFO_VALUEINT               35
#define WSM_INFO_VALUELABEL             36
#define WSM_INFO_MANDGROUPS             37
#define WSM_INFO_MCOMPS                 38
#define WSM_INFO_GROUPTYPE              39
#define WSM_INFO_OBJECTTYPE             40
#define WSM_INFO_WRITESYNTAX            41
#define WSM_INFO_WSYNTAXSIZES           42
#define WSM_INFO_WSYNTAXVALS            43
#define WSM_INFO_MINACCESS              44
#define WSM_INFO_SUPPORT                45
#define WSM_INFO_INCLUDES               46
#define WSM_INFO_VARIABLES              47
#define WSM_INFO_ACCESS                 48
#define WSM_INFO_CREATIONS              49
#define WSM_INFO_MODULE                 50

/* Node types */
#define WSM_MODULE_IDENTITY             1
#define WSM_OBJECT_IDENTITY             2
#define WSM_OBJECT_TYPE                 3
#define WSM_NOTIFICATION_TYPE           4
#define WSM_OBJECT_GROUP                5
#define WSM_MODULE_COMPLIANCE           6
#define WSM_AGENT_CAPABILITIES          7
```

```
/* ListElement types */
#define WSM_REV_LIST                    20
#define WSM_INDEX_LIST                  21
#define WSM_SIZE_LIST                   22
#define WSM_VALUE_LIST                  23
#define WSM_OBJECT_LIST                 24
#define WSM_MCMOD_LIST                  25
#define WSM_MCCOMP_LIST                 26
#define WSM_ACMOD_LIST                  27
#define WSM_ACVAR_LIST                  28

/* function prototypes: IN and OUT are for information purposes only */
#define WSM_CALL        WINAPI      /* for PASCAL calling conventions */

HWSM_NODE WSM_CALL WsmMapStrToNode
                (IN  LPSTR                  nodeString);
HWSM_NODE WSM_CALL WsmMapOidToNode
                (IN  smiLPOID               oid);
HWSM_NODE WSM_CALL WsmFindNodeChild
                (IN  HWSM_NODE              hNode);
HWSM_NODE WSM_CALL WsmFindNodeParent
                (IN  HWSM_NODE              hNode);
HWSM_NODE WSM_CALL WsmFindNodeNextSibling
                (IN  HWSM_NODE              hNode);
HWSM_NODE WSM_CALL WsmFindNodePrevSibling
                (IN  HWSM_NODE              hNode);
HWSM_LISTELEMENT WSM_CALL WsmFindNextListElement
                (IN  HWSM_LISTELEMENT       hListElement);
SNMPAPI_STATUS WSM_CALL WsmGetNodeOid
                (IN  HWSM_NODE              hNode,
                 IN  smiINT                 infoID,
                 OUT smiLPOID               oid);
SNMPAPI_STATUS WSM_CALL WsmGetNodeInt
                (IN  HWSM_NODE              hNode,
                 IN  smiINT                 infoID,
                 OUT smiLPINT               lpInt);
SNMPAPI_STATUS WSM_CALL WsmGetNodeString
                (IN  HWSM_NODE              hNode,
                 IN  smiINT                 infoID,
                 IN OUT smiLPINT            lpStrlen,
                 OUT LPSTR                  string);
SNMPAPI_STATUS WSM_CALL WsmGetNodeList
                (IN  HWSM_NODE              hNode,
                 IN  smiINT                 infoID,
                 OUT LPHWSM_LISTELEMENT     lphListElement);
SNMPAPI_STATUS WSM_CALL WsmGetListElementOid
                (IN  HWSM_LISTELEMENT       hListElement,
                 IN  smiINT                 infoID,
                 OUT smiLPOID               oid);
SNMPAPI_STATUS WSM_CALL WsmGetListElementInt
                (IN  HWSM_LISTELEMENT       hListElement,
                 IN  smiINT                 infoID,
                 OUT smiLPINT               lpInt);
SNMPAPI_STATUS WSM_CALL WsmGetListElementString
                (IN  HWSM_LISTELEMENT       hListElement,
```

```
                    IN  smiINT                  infoID,
                    IN OUT smiLPINT             lpStrlen,
                    OUT LPSTR                   string);
SNMPAPI_STATUS WSM_CALL WsmGetListElementList
                   (IN  HWSM_LISTELEMENT    hListElement,
                    IN  smiINT              infoID,
                    OUT LPHWSM_LISTELEMENT  lphListElement);
SNMPAPI_STATUS WSM_CALL WsmGetLastError
                   (void);
SNMPAPI_STATUS WSM_CALL WsmFreeList
                   (HWSM_LISTELEMENT        hListElement);
SNMPAPI_STATUS WSM_CALL WsmLoadModule
                   (LPSTR                   module);
SNMPAPI_STATUS WSM_CALL WsmUnloadModule
                   (LPSTR                   module);


#endif   /* _INC_WSNMPMIB  */
```