CONFIDENTIAL

Application Note 4
Sensor Node Software

Adding New Objects to the *GainSpan Sensor Platform Software* Management Information Base (MIB)

INTRODUCTION

NETWORK MANAGEMENT SYSTEMS (NMS) use the Simple Network Management Protocol (SNMP) to manage and control network devices. Each managed device includes a database of managed objects called a Management Information Base (MIB) and an SNMP agent. The MIB is hierarchically-structured (tree-shaped) database of managed objects, defined using ASN.1 notation. This tree has the individually managed objects as its leaves. Several IETF RFCs define the root and first few levels of the standard MIB hierarchy. A numeric tag called the object identifier (OID) uniquely identifies each managed object in the MIB. The NMS accesses these objects by issuing SNMP SET and GET requests to the agent, specifying its OID. The agent can also notify the NMS of significant events by sending unsolicited messages, called SNMP TRAPs.

The *GainSpan Sensor Platform Software* (*GSPS*) includes an SNMP agent that is an add-on to its Green Hills Software (GHS) GHNet network stack, both of which were originally developed by Treck, Inc. Please refer to the *SNMP User Guide* for detailed technical information about the GHS SNMP agent. The *GSPS* MIB is defined in the file `LPX-MIB`.

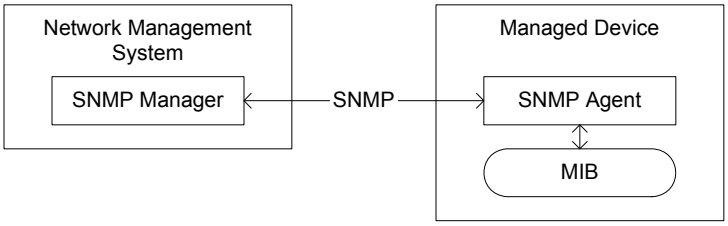

Figure 1: Network Management Systems use SNMP to manage network devices

The following is a link to a useful SNMP tutorial: http://www.dpstele.com/layers/l2/snmp_tutorials.php

PROCEDURES FOR ADDING MANAGED OBJECTS TO A MIB

There are two options for adding new application-specific objects to the standard *GSPS* MIB. They are both explained below.

Method 1: The Hard Way

This method can be used to add new, fully specified MIB objects anywhere in the MIB hierarchy.

1. Find a spot for the new object in existing MIB

The first step to adding a new object to an existing MIB is to pick a place in the hierarchy for the new object. Each object has an OID that uniquely identifies it, but also explicitly specifies its location in the hierarchy. SNMP OIDs use a dotted decimal notation, where each digit represents which leaf node is specified at a given level in the hierarchy. Many levels of the SNMP hierarchy have already been well defined by several IETF RFCs. Vendor-specific MIBs all live starting at iso(1).org(3).dod(6).internet(1).private(4).enterprise(1). The Internet Assigned Numbers Association (IANA) maintains this level of the hierarchy. For example, GainSpan's vendor-specific OID is 1.3.6.1.4.1.28295. A complete up-to-date list of assigned Private Enterprise Numbers (PEN) can be found at http://www.iana.org/assignments/enterprise-numbers. Reserving a PEN for your own organization is very easy and completely free! Just register at http://pen.iana.org/pen/PenApplication.page.

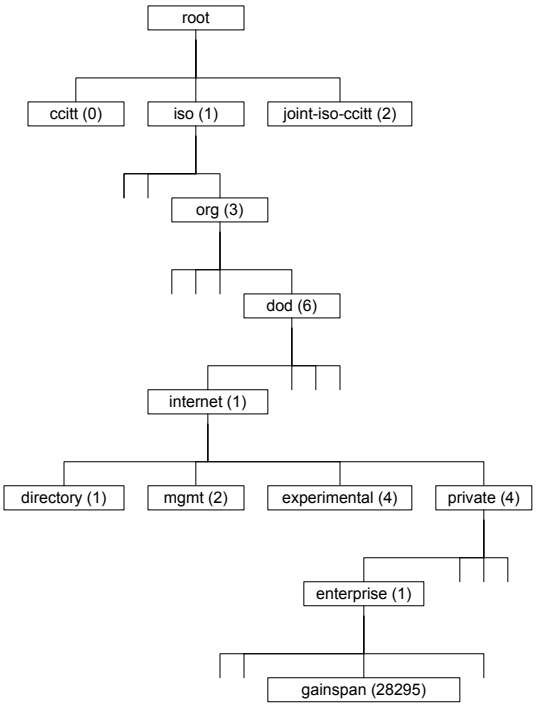

Figure 2: SNMP MIB OID hierarchy, showing GainSpan MIB OID.

The *GSPS* MIB is defined in the file `LPX-MIB`. This file starts by defining the GainSpan Enterprise MIB:

```
GAINSPAN-ENTERPRISE-MIB DEFINITIONS ::= BEGIN
```

Next, several externally-defined objects are imported from various public RFC MIB files.

```
IMPORTS
    IpAddress, Counter, Gauge, TimeTicks
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    TRAP-TYPE
        FROM RFC-1215;
```

This is followed by a description of the **?**

2. Define the New MIB Object and Add It to the Existing MIB

An SNMP MIB is defined using a subset of ASN.1 notation. ASN.1 defines some simple primitive data types (like integer), complex constructed data types, and macros. A set of Basic Encoding Rules (BER) is used to translate these abstract data values into a string of octets (8-bit bytes) that are actually sent over a network.

SNMP v1 uses these ASN.1 primitive data types:

Table 1: ASN.1 primitive types for SNMP (From http://www.et.put.poznan.pl/snmp/asn1/indexasn.html)

Type	*Description*
INTEGER	A whole number
OCTET STRING	A string of octets representing hexadecimal data
OBJECT IDENTIFIER	A string of numbers derived for a naming tree, used to identify an object
NULL	An empty placeholder
ENUMERATED	A limited set of integers with an assigned meaning
BOOLEAN	An integer with only two valid values: true = 1 and false = 2
COUNTER	*SNMP-specific.* An integer which increases up to a maximum value and the overflows to 0
GAUGE	*SNMP-specific.* An integer which increases up to a maximum and then decreases back to 0
TIMETICKS	
IPADDRESS	A four-octet string of hexadecimal data
NETWORKADDRESS	

SNMP also uses constructors, for building more complex data structures from primitive types.

Type	*Description*
SEQUENCE	An ordered list of datatypes
SEQUENCE OF	An ordered list of the same datatype

For example, the following definition shows a resource data type which is made up of three fields: an INTEGER, an OCTET STRING, and a BOOLEAN:

```
Resource ::=
   SEQUENCE {
        Id   INTEGER,
        Name OCTET STRING,
        Busy BOOLEAN,
   }
```

SNMP MIB objects are defined using an ASN.1 OBJECT-TYPE macro template. This template specifies an object's name, its OID, data type (complex or simple), value range, allowed operations (access), and descriptive information about the object.

The following example defines a new MIB object, called newMibObject. It is as an integer, both readable and writeable, and must be present in the MIB hierarchy. The purpose of the object is noted by its description "A new MIB Object". Lastly, assuming that this object's parent node, newMibRootObject has OID 1.3.6.1.4.999.3, this new object will be located at 1.3.6.1.4.999.3.9:

```
newMibObject OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
            "A new MIB Object"
     ::= { newMibRootObject 9 }
```

3. Compile New MIB

GHS provides a MIB compiler, *smidump*, which converts this human readable format to C-language header and source files, which can then be compiled and linked with application software.

At a command prompt, set the working directory to that which contains the `LPX-MIB` file and execute the following command to compile the MIB file:

```
C:> smidump –f Treck .\LPX-MIB
```

The argument "`–f Treck`" specifies to generate C source and header files compatible with the Treck (GHNet) SNMP agent. The "`.\LPX-MIB`" argument specifies to use the *LPX-MIB* module in the current directory. The command generates the following files:

- `gainspan-enterprise-mib_var.h`
- `gainspan-enterprise-mib_var.c`
- `gainspan-enterprise-mib_local.h`
- `gainspan-enterprise-mib_local.c`
- `gainspan-enterprise-mib.ins`

Copy these files to `$SNS_ROOT\GHS\ghnet2\snmpd\`, where `$SNS_ROOT` is the root directory of the Sensor Node Software source code distribution.

4. Modify MIB Compiler-generated Files

The newly generated file `gainspan-enterprise-mib.ins` contains instructions for modifying some of the source files in `$SNS_ROOT\GHS\ghnet2\snmpd\`. Follow these instructions carefully to hand-modify these files.

In order to fit the resulting built image into the available flash memory, please also modify the following files. To perform these modifications, please refer to the original GainSpan-provided SNMP files in `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan\:`.

- `trvars.h:`

  Copy all "`#ifdef WLAN_`" statements (and their corresponding `#endif` statements) from the GainSpan-provided `tvars.h` to the newly-generated `tvars.h`.

  For example, copy the line:

  ```
  #ifdef WLAN_MECFG_MIB_SUPPORT
  ```

  before

  ```
  {tfvar_mgmtenty,TM_SNMP_RWRITE,0,3,DESIREDBEACONPERIOD, … }
  ```

- `gainspan-enterprise-mib_var.c:`

  Copy all "`#ifdef WLAN_`" statements (and their corresponding `#endif` statements) from the GainSpan-provided `tvars.h` to the newly-generated *`tvars.h`*.

  For example, copy the line:

```
#ifdef WLAN_MECFG_MIB_SUPPORT
```

before

```
u_char *tfvar_mgmtenty(
```

- To further reduce the flash space required for the SNMP agent, insert "`#ifdef LPX_SNMP_ERROR_CHECK`" in all the same places as in the GainSpan-provided file.

5. Replace MIB Compiler-Generated MIB Instrumentation Files

The MIB compiler *smidump* generates a default "instrumentation" module. This module implements the handler stubs for all SNMP set, get and trap messages defined in the MIB. GainSpan supplies files that contain the handler implementation (located at `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan\`) for GainSpan-defined MIB objects that replace the MIB compiler generated stubs (located in `$SNS_ROOT\GHS\ghnet2\snmpd\`).

- Replace `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan-enterprise-mib_local.h` with `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan\gainspan-enterprise-mib_local.h`
- Replace `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan-enterprise-mib_local.c` with `$SNS_ROOT\GHS\ghnet2\snmpd\gainspan\gainspan-enterprise-mib_local.c`

6. Write Custom MIB Instrumentation Implementation

Handlers must also be defined for the new MIB objects. The best place for these handlers is in the sensor application module that will use the new MIB objects. On initialization, the application must call the API function LpxRegisterSnmpCb() to register these handlers with the SNMP agent. If the MIB handlers are not registered, the SNMP agent will ignore incoming MIB requests for what it perceives as unknown MIB objects.

- Define a handler function to process SNMP SET and GET messages bound for the newly-defined MIB objects.
- During application initialization, call LpxRegisterSnmpCb()to register the handler with the SNMP Agent. Please refer to the API documentation for more details about using this function.
- Once the handler is registered, the SNMP agent will call your function when it receives SNMP SET/GET commands for the MIB objects. The handler should then perform any required operations. Typically, this is reading or writing parameters to the flash parameter region reserved by the sensor application.

7. Rebuild Entire Project (*GSPS* and Application)

Both the GH Net SNMP agent and the sensor application must be recompiled for the new MIB to take effect. Once both modules have been compiled, the entire project can be built. Once built and downloaded to the device, performing SNMP SET/GET on the new MIB objects can test the newly added MIB objects.

Method 2: The Easy Way

It is possible to use SNMP for configuration without modifying the MIB definition file and rebuilding the *GainSpan Sensor Node Software*. GainSpan provides a vendor-specific MIB, which can be used to send up to 256 bytes of data defined in an application specific way. Please refer to the *GainSpan Sensor Node Software* API document for the MIB definition of the file.

To utilize the vendor-specific MIB, follow these steps:

- In the sensor application source code file (e.g., `LpxSensors\MySensor\MySensor.c`), define a set of handlers to process SNMP GET and SET messages for this MIB object. Typically these handlers would read from (for GET) or write to (SET) the flash configuration region. These functions should have the following prototypes:
  - `VOID MySensorGetCfg(SENSOR_CONFIG_PTR ConfigPtr)`
  - `VOID MySensorSetCfg(UINT8 Param, UINT8 * value, UINT32 length)`
- Register these as SNMP Callback functions using LpxRegisterSnmpCb() in a public function.
- Allocate enough bytes to the vendor-specific MIB for all configuration parameters. The vendor-specific MIB object (lpxvendorspecificdata) is defined as a string, and allows up to 256 bytes of storage. A second MIB object (lpxvendorspecificdata_len) stores the length of bytes allocated.
- Rebuild the application (`LpxAppSw.gpj`) only. It is not necessary to rebuild *GSPS*.

GainSpan Corporation • 121 Albright Way • Los Gatos, CA 94032-1801 • U.S.A.
+1 (408) 689-2129 • info@GainSpan.com • www.GainSpan.com

080127TE