PRECISE Software Technologies Inc.
THE RTOS SOLUTION COMPANY

Application Note

Adding a Management Information Base (MIB) to Precise/RTCS

This application note describes how to add a MIB (RFC 1155) to the Precise/RTCS embedded TCP/IP stack.

The process consists of writing a MIB definition file and compiling it to produce C source code.

The application note ends with an example of a definition file.

Writing a MIB definition file

A MIB definition file (**`.def`**) has two sections, separated by one separator line, as shown:

- Object-definition lines (one for each MIB object)
  Comment lines (optional)
  Type-renaming lines (optional)

%%

- C code

Object-definition lines

Each object definition is a single line with this format:

objectname parent.number **`[`** ***type access status*** **`[`** ***index index index*** **`...]]`**

Fields in the object definition are described in subsequent paragraphs.

Headquarters
Precise Software Technologies Inc.
301 Moodie Drive, Suite 308
Nepean ON Canada K2H 9C4
Telephone 613-596-2251
Facsimile 613-596-6713
Sales 800-265-9833

If the MIB object is:	Object definition includes:
Any object that is a nonleaf node	*objectname* *parent*.*number*
Any variable object	*objectname* *parent*.*number* *type* *access* *status*
A variable object that is a table row	*objectname* *parent*.*number* *type* *access* *status* *index* (row identifier; one for each of the table-row indices) For each index, you must include an object definition that defines the index as a variable object with the table entry as its parent.

type
One of:

- INTEGER
- OCTET (for OCTET STRING)
- OBJECT (for OBJECT IDENTIFIER)
- SEQUENCE (for SEQUENCE and SEQUENCE OF)
- IpAddress
- Counter
- Gauge
- TimeTicks
- Opaque

access
One of:

- read-only
- read-write
- write-only
- not-accessible

status
This field is ignored.

Examples

Object definition for the *system* subtree (object that is a non-leaf node)

```
system mib-2.1
```

system is child number 1 of node *mib-2*

Object definition for the *sysDescr* variable in the *system* subtree

```
sysDescr system.1 OCTET read-only mandatory
```

sysDescr is child number 1 of node *system*. It is a variable of type OCTET STRING, it is read only, and its implementation is mandatory.

Object definition for the *udpEntry* table entry

```
udpEntry udpTable.1 SEQUENCE not-accessible mandatory udpAddr udpPort
```

The line defines the format of an *udpEntry* entry in the *udpTable* table. The entry is indexed by variables *udpAddr* and *udpPort*.

You must include one object definition for *udpAddr* and one for *udpPort* than define each as a variable object with the *udpEntry* as its parent.

Comment lines

Lines that begin with **--** and have text on the same line are treated as comments by the MIB compiler:

```
-- This is a comment
```

Type-renaming lines

Lines that begin with **%%** rename types:

```
%% new_type existing_type
```

Separator line

A line that begins with **%%** and has no other text separates the object-definition section from the C-code section. The MIB compiler copies unchanged all lines following the separator line to its output C source file.

C code

As well as generating some other code, the MIB compiler generates and partially initializes variables of type RTCSMIB_NODE for each object that you define in the definition file.

For each:	C code must include:
MIB object	Code to initialize certain fields, as described in the following section
Root object in the definition file (a root object is one that does not have a parent defined in the definition file)	A call to: **`RTCSMIB_mib_add(&MIBNODE_`*objectname*`)`** (The call makes the object known to SNMP agent)
Table object	A function to map table indices to instances (*MIB_find_**objectname***(), described in a following section)
Writable variable object	A function to perform a set operation (*MIB_set_**objectname***(), described in a following section)

RTCSMIB_NODE variables

For each object that you define in the definition file, the MIB compiler generates a variable of type RTCSMIB_NODE and initializes the fields. However, you must write C code to override the following two fields, because the compiler has insufficient information to initialize them properly.

```
typedef struct rtcsmib_node {
   uint_32                    TYPE;    /* ignored in nonleaf nodes */
   union {
      uint_32                                  INT_CONST;
      uint_32_ptr                              INT_PTR;
      uint_32                   (_CODE_PTR_ INT_FN)(pointer);
      uchar_ptr                                DISPSTR_PTR;
      uchar_ptr                 (_CODE_PTR_ DISPSTR_FN)(pointer);
      uchar_ptr                 (_CODE_PTR_ OCTSTR_FN)(pointer, uint_32 _PTR_);
      struct rtcsmib_node _PTR_                OID_PTR;
      struct rtcsmib_node _PTR_ (_CODE_PTR_ OID_FN)(pointer);
   } GET;
} RTCSMIB_NODE, _PTR_ RTCSMIB_NODE_PTR;
```

If the variable object is a:	TYPE field is:	Initialize this GET field:	With:
Constant INTEGER	RTCSMIB_NODETYPE_ INT_CONST	INT_CONST	Value of variable object
INTEGER whose value is always at the same address	RTCSMIB_NODETYPE_ INT_PTR	INT_PTR	Address of variable object
INTEGER whose value SNMP agent computes when SNMP manager performs GET	RTCSMIB_NODETYPE_ INT_FN	INT_FN	Address of a function that takes an instance pointer (**`void *`**), returning value of variable object
NULL-terminated OCTET STRING whose value is always at the same address	RTCSMIB_NODETYPE_ DISPSTR_PTR	DISPSTR_PTR	Address of variable object
NULL-terminated OCTET STRING whose value SNMP agent computes when SNMP manager performs GET	RTCSMIB_NODETYPE_ DISPSTR_FN	DISPSTR_FN	Address of a function that takes an instance pointer (**`void *`**), returning address of a static buffer that contains value of variable object (must be static, because SNMP does not free it)

If the variable object is a:	TYPE field is:	Initialize this GET field:	With:
OCTET STRING whose value SNMP agent computes when SNMP manager performs GET	RTCSMIB_NODETYPE_ OCTSTR_FN	OCTSTR_FN	Address of a function that takes an instance pointer (**`void *`**), returning address of a static buffer that contains value and length of variable object (must be static, because SNMP does not free it)
Constant OBJECT ID	RTCSMIB_NODETYPE_ OID_PTR	OID_PTR	Address of an initialized RTCSMIB_NODE variable
Nonconstant OBJECT ID	RTCSMIB_NODETYPE_ OID_FN	OID_FN	Address of a function that takes an instance pointer (**`void *`**), returning address of an initialized RTCSMIB_NODE variable

In the preceding cases that initialize GET.*_FN fields, the field is a pointer to a function that takes an instance pointer as a parameter and that the SNMP agent calls to compute the value of a variable object.

For each variable object that is:	SNMP calls the appropriate GET.*_FN with:
Not in a table	A NULL instance pointer
Is in a table	The instance pointer from *MIB_find_**objectname***(), where *objectname* is the name of the variable object

You must provide *MIB_find_**objectname***(), as described in the following section.

MIB_find_*objectname*()

For each variable object that is in a table, you must provide *MIB_find_**objectname***(), where *objectname* is the name of the variable object. The function gets an instance pointer.

```
boolean MIB_find_objectname
(
   uint_32        op,       /* IN */
   pointer        index,    /* IN */
   pointer _PTR_  instance  /* OUT */
)
```

index

Pointer to a structure that contains the table index as described in the following table.

For each index of type:	The structure contains a field of this datatype:
Counter, Gauge, INTEGER, TimeTicks	*uint_32*
IpAddress	*uchar*`[4]`

The following table defines the returned values (TRUE/FALSE and ***instance*) according to the input values of *op* (the operation to be performed) and *index*. If the function returns FALSE, you must not assign a value to ***instance*.

For this input *op*:	Function returns TRUE if a table row exists with an index that:	If the function returns TRUE, assign ****instance* this value:
RTCSMIB_OP_GET	Corresponds to input *index*	If *_FN expects it, a pointer that SNMP manager passes as the instance pointer to any function that you assign to the GET union in RTCSMIB_NODE variables that correspond to entries in the table
RTCSMIB_OP_GETNEXT	Is lexicographically greater than input *index*	Same as above (Assign ***instance* the lexicographically greater index)
RTCSMIB_OP_SET	Corresponds to input *index* or, if not, can be created. If the table row does not exist, but can be created, the function must create the row.	A pointer that SNMP manager passes as the instance pointer to *MIB_set_**objectname***()

MIB_set_*objectname*()

For each writable variable object, you must provide *MIB_set_**objectname***(), where *objectname* is the name of the variable object.

```
uint_32 MIB_set_objectname
(
   pointer    instance,   /* IN */
   uchar_ptr  value_ptr,  /* OUT */
   uint_32    value_len   /* OUT */
)
```

instance
One of:

- NULL (if *objectname* is not in a table)
- Pointer returned by *MIB_find_objectname*()

value_ptr
Pointer to the value to which to set *objectname*.

value_len
Length in bytes of the value.

If *objectname* is an INTEGER (ASN.1 encoded), the following function is available so that you can parse *objectname*:

`RTCSMIB_int_read(`*`value_ptr, value_len`*`)`

Return codes

If the SNMP SET operation fails for this reason:	MIB_set_*objectname*() returns this code:
(The operation is successful)	SNMP_ERROR_noError
Value cannot be assigned to objectname because the value is illegal	SNMP_ERROR_wrongValue
Value is legal, but it cannot currently be assigned to objectname	SNMP_ERROR_inconsistentValue
value_len is incorrect for objectname type	SNMP_ERROR_wrongLength
There are not enough resources	SNMP_ERROR_resourceUnavailable
Any other reason	SNMP_ERROR_genErr

Compiling the MIB definition file

Accompanying Precise/RTCS is a script (**`postmosy.awk`**) that compiles a MIB definition file to C source code.

To compile the MIB definition file *mymib*`.def`, type:

`gawk -f postmosy.awk `*`mymib`*`.def > `*`mymib`*`.c`

By default, the utilities are installed in these directories:

```
gawk          c:\precise\mkbin
postmosy.awk  c:\precise\mkscript
```

The MIB compiler does the following and writes its output to its output C source file (*mymib*`.c`).

1. Generates a variable of type RTCSMIB_NODE for each object that you define in the MIB definition file. The variable is called MIBNODE_*objectname*.

   For example, for this object definition:

   ```
   system mib-2.1
   ```

   the MIB compiler generates this variable:

   ```
   MIBNODE_system
   ```

2. Generates functions that parse table indices.
3. Copies all lines that follow the `%%` separator line; that is, copies the C-code section of the definition file.

Compiling an ASN.1 grammar

Accompanying Precise/RTCS is a script (`mosy.awk`) that compiles an ASN.1 grammar to a MIB definition file.

To compile the ASN.1 grammar *mymib*`.txt`, type:

```
gawk -f mosy.awk mymib.txt > mymib.def
```

By default, the utilities are installed in these directories:

```
gawk            c:\precise\mkbin
mosy.awk        c:\precise\mkscript
```

Example: A MIB definition file

```
-- This definition file is a subset of RFC 1213

%% DisplayString OCTET

mib-2           mgmt.1
system          mib-2.1
udp             mib-2.7

sysDescr        system.1   DisplayString   read-only       mandatory
sysObjectID     system.2   OBJECT          read-only       mandatory
sysUpTime       system.3   TimeTicks       read-only       mandatory
sysContact      system.4   DisplayString   read-write      mandatory

udpInDatagrams  udp.1      Counter   read-only      mandatory
udpOutDatagrams udp.4      Counter   read-only      mandatory
udpTable        udp.5      SEQUENCE  not-accessible mandatory
udpEntry        udpTable.1 SEQUENCE  not-accessible mandatory udpLocalAddr
udpLocalPort
udpLocalAddr    udpEntry.1 IpAddress read-only      mandatory
udpLocalPort    udpEntry.2 INTEGER   read-only      mandatory

%%

extern RTCSMIB_NODE MIBNODE_mqx;
extern UDP_STATS UDP;
```

```
uint_32   gettime(pointer);
uchar_ptr getudpaddr(pointer, uint_32 _PTR_);
uint_32   getudpport(pointer);

void mib_init(void)
{ /* Body */

   MIBNODE_sysDescr.TYPE = RTCSMIB_NODETYPE_DISPSTR_PTR;
   MIBNODE_sysDescr.GET.DISPSTR_PTR = (uchar_ptr)"Precise/RTCS";

   MIBNODE_sysObjectID.TYPE = RTCSMIB_NODETYPE_OID_PTR;
   MIBNODE_sysObjectID.GET.OID_PTR = &MIBNODE_mqx;

   MIBNODE_sysUpTime.TYPE = RTCSMIB_NODETYPE_INT_FN;
   MIBNODE_sysUpTime.GET.INT_FN = gettime;

   MIBNODE_sysContact.TYPE = RTCSMIB_NODETYPE_DISPSTR_PTR;
   MIBNODE_sysContact.GET.DISPSTR_PTR = NULL;


   MIBNODE_udpInDatagrams.TYPE = RTCSMIB_NODETYPE_INT_PTR;
   MIBNODE_udpInDatagrams.GET.INT_PTR = &UDP.RX_TOTAL;

   MIBNODE_udpOutDatagrams.TYPE = RTCSMIB_NODETYPE_INT_PTR;
   MIBNODE_udpOutDatagrams.GET.INT_PTR = &UDP.TX_TOTAL;

   MIBNODE_udpLocalAddr.TYPE = RTCSMIB_NODETYPE_OCTSTR_FN;
   MIBNODE_udpLocalAddr.GET.OCTSTR_FN = getudpaddr;

   MIBNODE_udpLocalPort.TYPE = RTCSMIB_NODETYPE_INT_FN;
   MIBNODE_udpLocalPort.GET.INT_FN = getudpport;

   RTCSMIB_mib_add(&MIBNODE_mib2);
} /* Endbody */

uint_32 gettime
   (
      pointer   dummy   /* system.sysUpTime isn't in a table, so dummy is
                           always NULL */
   )
{ /* Body */

   TIME_STRUCT uptime;
   _time_get_elapsed(&uptime);
   return (uptime.SECONDS * 100) + (uptime.MILLISECONDS / 10);

} /* Endbody */

uint_32 MIB_set_sysContact
   (
      pointer   dummy,  /* system.sysContact isn't in a table, so dummy is
                           always NULL */
      uchar_ptr varptr,
      uint_32   varlen
   )
{ /* Body */
   uchar_ptr dispstr = NULL;
```

```
   if (varlen) {
      dispstr = _mem_alloc(varlen+1);
      if (!dispstr) {
         return SNMP_ERROR_resourceUnavailable;
      } /* Endif */
      _mem_copy(varptr, dispstr, varlen);
      dispstr[varlen] = '\0';
   } /* Endif */

   if (MIBNODE_sysContact.GET.DISPSTR_PTR) {
      _mem_free(MIBNODE_sysContact.GET.DISPSTR_PTR);
   } /* Endif */

   MIBNODE_sysContact.GET.DISPSTR_PTR = dispstr;
   return SNMP_ERROR_noError;

} /* Endbody */

boolean MIB_find_udpEntry
   (
      uint_32        op,
      pointer        index,
      pointer _PTR_  instance
   )
{ /* Body */
   UCB_STRUCT_PTR       ucb_ptr, search_ptr;
   _ip_address          locaddr;
   uint_16              locport;
   struct {
      uchar   udpLocalAddress[4];
      uint_32 udpLocalPort;
   } _PTR_ realindex = index;

   locaddr = ntohl(realindex->udpLocalAddress);
   locport = realindex->udpLocalPort;

   ucb_ptr = NULL;
   switch (op) {
   case RTCSMIB_OP_GET:
   case RTCSMIB_OP_SET:
      for (search_ptr = UCB_HEAD; search_ptr; search_ptr = search_ptr->NEXT) {
         if ((search_ptr->IPADDR == locaddr)
          && (search_ptr->PORT   == locport)) {
            ucb_ptr = search_ptr;
            break;
         } /* Endif */
      } /* Endfor */
      break;
```

```
   case RTCSMIB_OP_GETNEXT:
      for (search_ptr = UCB_HEAD; search_ptr; search_ptr = search_ptr->NEXT) {
         if ((search_ptr->IPADDR >  locaddr)
         || ((search_ptr->IPADDR == locaddr)
          && (search_ptr->PORT   >= locport))) {
            if (ucb_ptr == NULL) {
               ucb_ptr = search_ptr;
            } else if ((search_ptr->IPADDR <  ucb_ptr->IPADDR)
                   || ((search_ptr->IPADDR == ucb_ptr->IPADDR)
                    && (search_ptr->PORT   <  ucb_ptr->PORT))) {
               ucb_ptr = search_ptr;
            } /* Endif */
         } /* Endif */
      } /* Endfor */
      break;
   } /* Endswitch */

   if (!ucb_ptr) {
      return FALSE;
   } /* Endif */

   htonl(realindex->udpLocalAddress, ucb_ptr->IPADDR);
   realindex->udpLocalPort = ucb_ptr->PORT;
   *instance = ucb_ptr;
   return TRUE;

} /* Endbody */

uchar_ptr getudpaddr
   (
      pointer       instance,   /* UCB_STRUCT_PTR returned by
                                   MIB_find_udpEntry */
      uint_32 _PTR_ len
   )
{ /* Body */
   UCB_STRUCT_PTR ucb_ptr = instance;
   static uchar udpaddr[4];

   htonl(udpaddr, ucb_ptr->IPADDR);
   *len = 4;
   return udpaddr;
} /* Endbody */

uint_32 getudpport
   (
      pointer       instance    /* UCB_STRUCT_PTR returned by
                                   MIB_find_udpEntry */
   )
{ /* Body */
   UCB_STRUCT_PTR ucb_ptr = instance;

   return ucb_ptr->PORT;
} /* Endbody */
```

www.psti.com