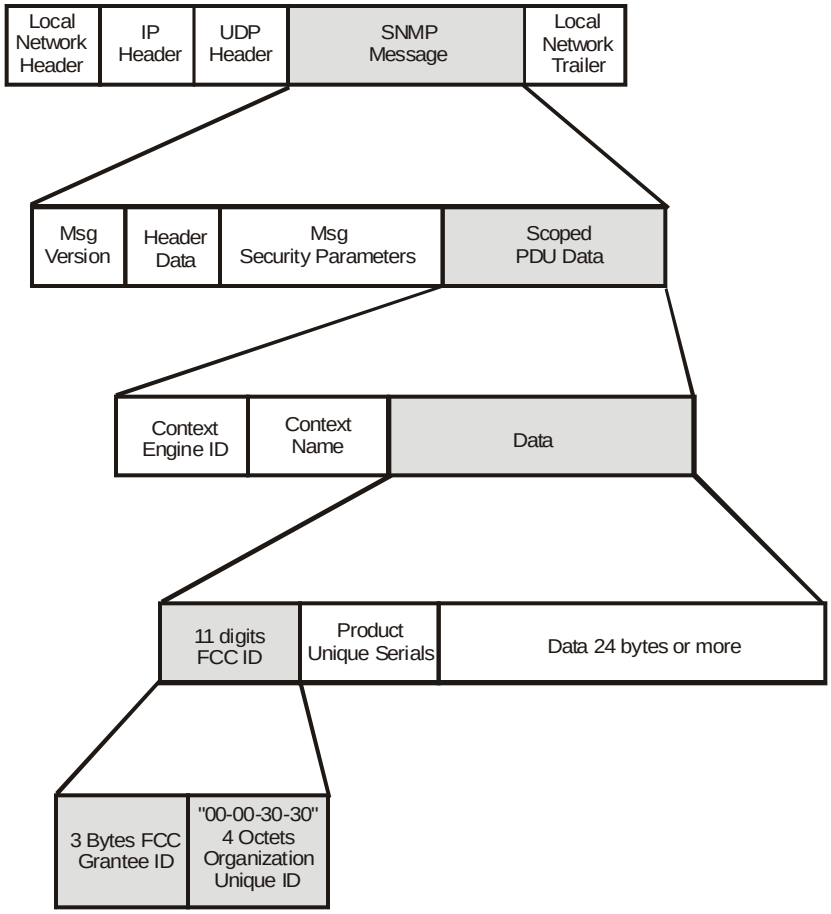
FCC Identifier

The FCC identifier is embedded as a part of maintenance packets/cell in GigaLink Systems. This way, the transmission of FCC identifier becomes transparent to the data communication over the radio systems and allows the user to monitor the health of radio devices that make up the network from a central location.
Specifically, the FCC Identifier is embedded in SNMP packets.
Thus, the FCC identifier along with device serial number and the status of each radio on the network are received and retransmitted by all the radios within the network. The same serial number is transmitted by each radio on the same network. The attached letter from Karen Rackley of the FCC states that the commission's rules do not require the transmission of a unique manufacture's serial number for each unlicensed device that operates in the 59-64GHz band, so that one serial number can identify all transmitters operation as part of a single communications system, see attached paper.

FCC ID in SNMP Packet format



Sample code to view SNMP packet.

```
/* v1.1 WinSNMP.h */
/* v1.0 - Sep 13, 1993 */
/* v1.1 - Jun 10, 1994 */

#ifndef _INC_WINSNMP /* Include WinSNMP declarations */
#define  _INC_WINSNMP          /* Just once! */

#ifndef _INC_WINDOWS           /* Include Windows declarations, if not already done */
#include <windows.h>
#define _INC_WINDOWS           /* Just once! */
#endif                         /* _INC_WINDOWS */

#ifdef __cplusplus
extern "C" {
#endif

/* WinSNMP API Type Definitions  */
typedef  HANDLE                HSNMP_SESSION,      FAR *LPHSNMP_SESSION;
typedef  HANDLE                HSNMP_ENTITY,       FAR *LPHSNMP_ENTITY;
typedef  HANDLE                HSNMP_CONTEXT,      FAR *LPHSNMP_CONTEXT;
typedef  HANDLE                HSNMP_PDU,          FAR *LPHSNMP_PDU;
typedef  HANDLE                HSNMP_VBL,          FAR *LPHSNMP_VBL;
typedef unsigned char          smiBYTE,            FAR *smiLPBYTE;
/* SNMP-related types from RFC1442 (SMI) */
typedef signed long            smiINT,             FAR *smiLPINT;
typedef smiINT                 smiINT32,           FAR *smiLPINT32;
typedef unsigned long          smiUINT32,          FAR *smiLPUINT32;
typedef struct {
    smiUINT32 len;
    smiLPBYTE ptr;}            smiOCTETS,          FAR *smiLPOCTETS;
typedef const smiOCTETS                            FAR *smiLPCOCTETS;
typedef smiOCTETS              smiBITS,            FAR *smiLPBITS;
typedef struct {
    smiUINT32   len;
    smiLPUINT32 ptr;}          smiOID,          FAR *smiLPOID;
typedef const smiOID                               FAR *smiLPCOID;
typedef smiOCTETS              smiIPADDR,          FAR *smiLPIPADDR;
typedef smiUINT32              smiCNTR32,          FAR *smiLPCNTR32;
typedef smiUINT32              smiGAUGE32,         FAR *smiLPGAUGE32;
typedef smiUINT32              smiTIMETICKS, FAR *smiLPTIMETICKS;
typedef smiOCTETS              smiOPAQUE,          FAR *smiLPOPAQUE;
typedef smiOCTETS              smiNSAPADDR, FAR *smiLPNSAPADDR;
typedef struct {
      smiUINT32 hipart;
      smiUINT32 lopart;}   smiCNTR64,             FAR *smiLPCNTR64;

/* ASN/BER Base Types */
/* (used in forming SYNTAXes and certain SNMP types/values) */
#define ASN_UNIVERSAL                (0x00)
#define  ASN_APPLICATION             (0x40)
#define  ASN_CONTEXT           (0x80)
#define  ASN_PRIVATE                 (0xC0)
#define  ASN_PRIMITIVE               (0x00)
#define  ASN_CONSTRUCTOR             (0x20)
```

```
/* SNMP ObjectSyntax Values */
#define  SNMP_SYNTAX_SEQUENCE     (ASN_UNIVERSAL | ASN_CONSTRUCTOR | 0x10)
/* These values are used in the "syntax" member of the smiVALUE structure which follows */
#define SNMP_SYNTAX_INT              (ASN_UNIVERSAL | ASN_PRIMITIVE | 0x02)
#define SNMP_SYNTAX_BITS             (ASN_UNIVERSAL | ASN_PRIMITIVE | 0x03)
#define SNMP_SYNTAX_OCTETS           (ASN_UNIVERSAL | ASN_PRIMITIVE | 0x04)
#define SNMP_SYNTAX_NULL             (ASN_UNIVERSAL | ASN_PRIMITIVE | 0x05)
#define SNMP_SYNTAX_OID              (ASN_UNIVERSAL | ASN_PRIMITIVE | 0x06)
#define SNMP_SYNTAX_INT32            SNMP_SYNTAX_INT
#define SNMP_SYNTAX_IPADDR           (ASN_APPLICATION | ASN_PRIMITIVE | 0x00)
#define SNMP_SYNTAX_CNTR32           (ASN_APPLICATION | ASN_PRIMITIVE | 0x01)
#define SNMP_SYNTAX_GAUGE32          (ASN_APPLICATION | ASN_PRIMITIVE | 0x02)
#define SNMP_SYNTAX_TIMETICKS        (ASN_APPLICATION | ASN_PRIMITIVE | 0x03)
#define SNMP_SYNTAX_OPAQUE           (ASN_APPLICATION | ASN_PRIMITIVE | 0x04)
#define SNMP_SYNTAX_NSAPADDR         (ASN_APPLICATION | ASN_PRIMITIVE | 0x05)
#define SNMP_SYNTAX_CNTR64           (ASN_APPLICATION | ASN_PRIMITIVE | 0x06)
#define SNMP_SYNTAX_UINT32           (ASN_APPLICATION | ASN_PRIMITIVE | 0x07)
/* Exception conditions in response PDUs for SNMPv2 */
#define SNMP_SYNTAX_NOSUCHOBJECT     (ASN_CONTEXT | ASN_PRIMITIVE | 0x00)
#define SNMP_SYNTAX_NOSUCHINSTANCE   (ASN_CONTEXT | ASN_PRIMITIVE | 0x01)
#define SNMP_SYNTAX_ENDOFMIBVIEW     (ASN_CONTEXT | ASN_PRIMITIVE | 0x02)

typedef struct {                            /* smiVALUE portion of VarBind */
        smiUINT32       syntax;             /* Insert SNMP_SYNTAX_<type> */
        union {
        smiINT          sNumber;            /* SNMP_SYNTAX_INT
                                               SNMP_SYNTAX_INT32 */
        smiUINT32       uNumber;            /* SNMP_SYNTAX_UINT32
                                               SNMP_SYNTAX_CNTR32
                                               SNMP_SYNTAX_GAUGE32
                                               SNMP_SYNTAX_TIMETICKS */
        smiCNTR64       hNumber;            /* SNMP_SYNTAX_CNTR64 */
        smiOCTETS       string;             /* SNMP_SYNTAX_OCTETS
                                               SNMP_SYNTAX_BITS
                                               SNMP_SYNTAX_OPAQUE
                                               SNMP_SYNTAX_IPADDR
                                               SNMP_SYNTAX_NSAPADDR */
        smiOID          oid;                /* SNMP_SYNTAX_OID */
        smiBYTE         empty;              /* SNMP_SYNTAX_NULL
                                               SNMP_SYNTAX_NOSUCHOBJECT
                                               SNMP_SYNTAX_NOSUCHINSTANCE
                                               SNMP_SYNTAX_ENDOFMIBVIEW */
            }           value;              /* union */
        }               smiVALUE,       FAR *smiLPVALUE;
typedef const smiVALUE          FAR *smiLPCVALUE;

/* SNMP Limits  */
#define MAXOBJIDSIZE 128        /* Max number of components in an OID */
#define MAXOBJIDSTRSIZE         1408    /* Max len of decoded MAXOBJIDSIZE OID */

/* PDU Type Values */
#define SNMP_PDU_GET                (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x0)
#define SNMP_PDU_GETNEXT    (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x1)
#define SNMP_PDU_RESPONSE   (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x2)
#define SNMP_PDU_SET            (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x3)
/* SNMP_PDU_V1TRAP is obsolete in SNMPv2 */
#define SNMP_PDU_V1TRAP             (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x4)
#define SNMP_PDU_GETBULK    (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x5)
#define SNMP_PDU_INFORM             (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x6)
#define SNMP_PDU_TRAP               (ASN_CONTEXT | ASN_CONSTRUCTOR | 0x7)
```

```
/* SNMPv1 Trap Values */
/* (These values might be superfluous wrt WinSNMP applications) */
#define SNMP_TRAP_COLDSTART                     0
#define SNMP_TRAP_WARMSTART                     1
#define SNMP_TRAP_LINKDOWN                      2
#define SNMP_TRAP_LINKUP                        3
#define SNMP_TRAP_AUTHFAIL                      4
#define SNMP_TRAP_EGPNEIGHBORLOSS               5
#define SNMP_TRAP_ENTERPRISESPECIFIC            6

/* SNMP Error Codes Returned in Error_status Field of PDU */
/* (these are NOT WinSNMP API Error Codes */
/* Error Codes Common to SNMPv1 and SNMPv2 */
#define SNMP_ERROR_NOERROR                      0
#define SNMP_ERROR_TOOBIG               1
#define SNMP_ERROR_NOSUCHNAME           2
#define SNMP_ERROR_BADVALUE                     3
#define SNMP_ERROR_READONLY                     4
#define SNMP_ERROR_GENERR               5
/* Error Codes Added for SNMPv2 */
#define SNMP_ERROR_NOACCESS                     6
#define SNMP_ERROR_WRONGTYPE                    7
#define SNMP_ERROR_WRONGLENGTH                  8
#define SNMP_ERROR_WRONGENCODING                9
#define SNMP_ERROR_WRONGVALUE           10
#define SNMP_ERROR_NOCREATION                   11
#define SNMP_ERROR_INCONSISTENTVALUE    12
#define SNMP_ERROR_RESOURCEUNAVAILABLE          13
#define SNMP_ERROR_COMMITFAILED         14
#define SNMP_ERROR_UNDOFAILED                   15
#define SNMP_ERROR_AUTHORIZATIONERROR           16
#define SNMP_ERROR_NOTWRITABLE          17
#define SNMP_ERROR_INCONSISTENTNAME     18

/* WinSNMP API Values */
/* Values used to indicate entity/context translation modes */
#define SNMPAPI_TRANSLATED          0
#define SNMPAPI_UNTRANSLATED_V1 1
#define SNMPAPI_UNTRANSLATED_V2 2

/* Values used to indicate SNMP "communications  level" supported by the implementation */
#define SNMPAPI_NO_SUPPORT              0
#define SNMPAPI_V1_SUPPORT          1
#define SNMPAPI_V2_SUPPORT          2
#define SNMPAPI_M2M_SUPPORT             3

/* Values used to indicate retransmit mode in the implementation */
#define SNMPAPI_OFF                             0       /* Refuse support */
#define SNMPAPI_ON                              1       /* Request support */

/* WinSNMP API Function Return Codes */
typedef smiUINT32               SNMPAPI_STATUS;     /* Used for function ret values */
#define SNMPAPI_FAILURE                 0       /* Generic error code */
#define SNMPAPI_SUCCESS                 1       /* Generic success code */
```

```
/* WinSNMP API Error Codes (for SnmpGetLastError) */
/* (NOT SNMP Response-PDU error_status codes) */
#define SNMPAPI_ALLOC_ERROR             2       /* Error allocating memory */
#define SNMPAPI_CONTEXT_INVALID         3       /* Invalid context parameter */
#define SNMPAPI_CONTEXT_UNKNOWN         4       /* Unknown context parameter */
#define SNMPAPI_ENTITY_INVALID          5       /* Invalid entity parameter */
#define SNMPAPI_ENTITY_UNKNOWN  6       /* Unknown entity parameter */
#define SNMPAPI_INDEX_INVALID           7       /* Invalid VBL index parameter */
#define SNMPAPI_NOOP                    8       /* No operation performed */
#define SNMPAPI_OID_INVALID             9       /* Invalid OID parameter */
#define SNMPAPI_OPERATION_INVALID       10      /* Invalid/unsupported operation */
#define SNMPAPI_OUTPUT_TRUNCATED        11      /* Insufficient output buf len */
#define SNMPAPI_PDU_INVALID             12      /* Invalid PDU parameter */
#define SNMPAPI_SESSION_INVALID         13      /* Invalid session parameter */
#define SNMPAPI_SYNTAX_INVALID          14      /* Invalid syntax in smiVALUE */
#define SNMPAPI_VBL_INVALID             15      /* Invalid VBL parameter */
#define SNMPAPI_MODE_INVALID            16      /* Invalid mode parameter */
#define SNMPAPI_SIZE_INVALID            17      /* Invalid size/length parameter */
#define SNMPAPI_NOT_INITIALIZED         18      /* SnmpStartup failed/not called */
#define SNMPAPI_MESSAGE_INVALID  19     /* Invalid SNMP message format */
#define SNMPAPI_HWND_INVALID            20      /* Invalid Window handle */
#define SNMPAPI_OTHER_ERROR             99      /* For internal/undefined errors */
/* Generic Transport Layer (TL) Errors */
#define SNMPAPI_TL_NOT_INITIALIZED      100     /* TL not initialized */
#define SNMPAPI_TL_NOT_SUPPORTED        101     /* TL does not support protocol */
#define SNMPAPI_TL_NOT_AVAILABLE        102     /* Network subsystem has failed */
#define SNMPAPI_TL_RESOURCE_ERROR       103     /* TL resource error */
#define SNMPAPI_TL_UNDELIVERABLE        104     /* Destination unreachable */
#define SNMPAPI_TL_SRC_INVALID          105     /* Source endpoint invalid */
#define SNMPAPI_TL_INVALID_PARAM        106     /* Input parameter invalid */
#define SNMPAPI_TL_IN_USE               107     /* Source endpoint in use */
#define SNMPAPI_TL_TIMEOUT      108     /* No response before timeout */
#define SNMPAPI_TL_PDU_TOO_BIG          109     /* PDU too big for send/receive */
#define SNMPAPI_TL_OTHER                199     /* Undefined TL error */

/* WinSNMP API Function Prototypes */
#define IN                              /* Documentation only */
#define OUT                             /* Documentation only */
#define SNMPAPI_CALL WINAPI             /* FAR PASCAL calling conventions */
```

/* Local Database Functions */

```
SNMPAPI_STATUS      SNMPAPI_CALL      SnmpGetTranslateMode
                                      (OUT smiLPUINT32 nTranslateMode);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpSetTranslateMode
                                      (IN smiUINT32 nTranslateMode);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpGetRetransmitMode
                                      (OUT smiLPUINT32 nRetransmitMode);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpSetRetransmitMode
                                      (IN smiUINT32 nRetransmitMode);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpGetTimeout
                                      (IN HSNMP_ENTITY hEntity,
                                      OUT smiLPTIMETICKS nPolicyTimeout,
                                      OUT smiLPTIMETICKS nActualTimeout);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpSetTimeout
                                      (IN HSNMP_ENTITY hEntity,
                                      IN smiTIMETICKS nPolicyTimeout);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpGetRetry
                                      (IN HSNMP_ENTITY hEntity,
                                      OUT smiLPUINT32 nPolicyRetry,
                                      OUT smiLPUINT32 nActualRetry);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpSetRetry
                                      (IN HSNMP_ENTITY hEntity,
                                      IN smiUINT32 nPolicyRetry);
```

```
/* Communications Functions */

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpStartup
                                      (OUT smiLPUINT32 nMajorVersion,
                                      OUT smiLPUINT32 nMinorVersion,
                                      OUT smiLPUINT32 nLevel,
                                      OUT smiLPUINT32 nTranslateMode,
                                      OUT smiLPUINT32 nRetransmitMode);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpCleanup
                                      (void);

HSNMP_SESSION  SNMPAPI_CALL      SnmpOpen
                                      (IN HWND hWnd,
                                      IN UINT wMsg);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpClose
                                      (IN HSNMP_SESSION session);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpSendMsg
                                      (IN HSNMP_SESSION session,
                                      IN HSNMP_ENTITY srcEntity,
                                      IN HSNMP_ENTITY dstEntity,
                                      IN HSNMP_CONTEXT context,
                                      IN HSNMP_PDU PDU);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpRecvMsg
                                      (IN HSNMP_SESSION session,
                                      OUT LPHSNMP_ENTITY srcEntity,
                                      OUT LPHSNMP_ENTITY dstEntity,
                                      OUT LPHSNMP_CONTEXT  context,
                                      OUT LPHSNMP_PDU PDU);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpRegister
                                      (IN HSNMP_SESSION session,
                                      IN HSNMP_ENTITY srcEntity,
                                      IN HSNMP_ENTITY dstEntity,
                                      IN HSNMP_CONTEXT context,
                                      IN smiLPCOID notification,
                                      IN smiUINT32 state);
```

```
/* Entity/Context Functions */

HSNMP_ENTITY        SNMPAPI_CALL        SnmpStrToEntity
                                        (IN HSNMP_SESSION session,
                                        IN LPCSTR string);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpEntityToStr
                                        (IN HSNMP_ENTITY entity,
                                        IN smiUINT32 size,
                                        OUT LPSTR string);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpFreeEntity
                                        (IN HSNMP_ENTITY entity);

HSNMP_CONTEXT       SNMPAPI_CALL        SnmpStrToContext
                                        (IN HSNMP_SESSION session,
                                        IN smiLPCOCTETS string);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpContextToStr
                                        (IN HSNMP_CONTEXT context,
                                        OUT smiLPOCTETS string);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpFreeContext
                                        (IN HSNMP_CONTEXT context);
```

```
/* PDU Functions */

HSNMP_PDU           SNMPAPI_CALL        SnmpCreatePdu
                                        (IN HSNMP_SESSION session,
                                        IN smiINT PDU_type,
                                        IN smiINT32 request_id,
                                        IN smiINT error_status,
                                        IN smiINT error_index,
                                        IN HSNMP_VBL varbindlist);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpGetPduData
                                        (IN HSNMP_PDU PDU,
                                        OUT smiLPINT PDU_type,
                                        OUT smiLPINT32 request_id,
                                        OUT smiLPINT error_status,
                                        OUT smiLPINT error_index,
                                        OUT LPHSNMP_VBL varbindlist);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpSetPduData
                                        (IN HSNMP_PDU PDU,
                                        IN const smiINT FAR *PDU_type,
                                        IN const smiINT32 FAR *request_id,
                                        IN const smiINT FAR *non_repeaters,
                                        IN const smiINT FAR *max_repetitions,
                                        IN const HSNMP_VBL FAR *varbindlist);

HSNMP_PDU           SNMPAPI_CALL        SnmpDuplicatePdu
                                        (IN HSNMP_SESSION session,
                                        IN HSNMP_PDU PDU);

SNMPAPI_STATUS      SNMPAPI_CALL        SnmpFreePdu
                                        (IN HSNMP_PDU PDU);
```

```
/* Variable-Binding Functions */

HSNMP_VBL           SNMPAPI_CALL    SnmpCreateVbl
                                    (IN HSNMP_SESSION session,
                                    IN smiLPCOID name,
                                    IN smiLPCVALUE value);

HSNMP_VBL           SNMPAPI_CALL    SnmpDuplicateVbl
                                    (IN HSNMP_SESSION session,
                                    IN HSNMP_VBL vbl);

SNMPAPI_STATUS      SNMPAPI_CALL    SnmpFreeVbl
                                    (IN HSNMP_VBL vbl);

SNMPAPI_STATUS      SNMPAPI_CALL    SnmpCountVbl
                                    (IN HSNMP_VBL vbl);

SNMPAPI_STATUS      SNMPAPI_CALL    SnmpGetVb
                                    (IN HSNMP_VBL vbl,
                                    IN smiUINT32 index,
                                    OUT smiLPOID name,
                                    OUT smiLPVALUE value);

SNMPAPI_STATUS      SNMPAPI_CALL    SnmpSetVb
                                    (IN HSNMP_VBL vbl,
                                    IN smiUINT32 index,
                                    IN smiLPCOID name,
                                    IN smiLPCVALUE value);

SNMPAPI_STATUS      SNMPAPI_CALL    SnmpDeleteVb
                                    (IN HSNMP_VBL vbl,
                                    IN smiUINT32 index);
```

```
/* Utility Functions */

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpGetLastError
                                      (IN HSNMP_SESSION session);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpStrToOid
                                      (IN LPCSTR string,
                                      OUT smiLPOID dstOID);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpOidToStr
                                      (IN smiLPCOID srcOID,
                                      IN smiUINT32 size,
                                      OUT LPSTR string);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpOidCopy
                                      (IN smiLPCOID srcOID,
                                      OUT smiLPOID dstOID);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpOidCompare
                                      (IN smiLPCOID xOID,
                                      IN smiLPCOID yOID,
                                      IN smiUINT32 maxlen,
                                      OUT smiLPINT result);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpEncodeMsg
                                      (IN HSNMP_SESSION session,
                                      IN HSNMP_ENTITY srcEntity,
                                      IN HSNMP_ENTITY dstEntity,
                                      IN HSNMP_CONTEXT context,
                                      IN HSNMP_PDU pdu,
                                      OUT smiLPOCTETS msgBufDesc);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpDecodeMsg
                                      (IN HSNMP_SESSION session,
                                      OUT LPHSNMP_ENTITY srcEntity,
                                      OUT LPHSNMP_ENTITY dstEntity,
                                      OUT LPHSNMP_CONTEXT context,
                                      OUT LPHSNMP_PDU pdu,
                                      IN smiLPCOCTETS msgBufDesc);

SNMPAPI_STATUS      SNMPAPI_CALL      SnmpFreeDescriptor
                                      (IN smiUINT32 syntax,
                                      IN smiLPOPAQUE descriptor);

#ifdef __cplusplus
}
#endif
#endif              /* _INC_WINSNMP */
```

```
SNMPAPI_STATUS TrapProcess (HSNMP_SESSION hSession)
{
    HSNMP_ENTITY hSrc, hDest;
    HSNMP_CONTEXT hContext;
    HSNMP_PDU hPDU;
    HSNMP_VBL hVBL;
    smiINT32 Request_id;
    smiINT PduType, Err_stat, Err_index;
    SNMPAPI_STATUS RetStatus, Index, VBCount;
    smiOID   Name;
    smiVALUE Value;
    smiBYTE  NameBuffer[100], ValueBuffer[256];
    ...
    Request_id = SnmpRecvMsg (
        hSession,                   // Trap Session Handle
        &hSrc,          // Source Entity Handle
        &hDest,                     // Destination Entity Handle
        &hContext,                  // Context Handle
        &hPDU);                     // PDU Handle
    // Error condition checking for SnmpRecvMsg() performs here.

    RetStatus = SnmpGetPduData (
        hPDU,           // PDU Handle
        &PduType,                   // PDU return type
        &Request_id,    // Request ID of the Trap
        &Err_stat,                  // Error status for a variable
        &Err_index,                 // Index to the variable with error
        &hVBL);                     // Handle to the Varbindlist

    // Sample error checking for SnmpGetPduData():
    if ((RetStatus == SNMPAPI_FAILURE) ||
       (PduType != SNMP_PDU_TRAP) ||
       (Err_stat != SNMP_ERROR_NOERROR)) {
        SnmpFreePdu(hPDU);
        SnmpFreeEntity(hSrc);
        SnmpFreeEntity(hDest);
        SnmpFreeContext(hContext);
        return (SnmpGetLastError(hSession));
    }

    VBCount = SnmpCountVbl(hVBL);
    for (Index = 1; i <= VBCount; Index ++) {

        // When Index = 1,
        //  Oid   = sysUpTimeOid
        //  Value = uptime value for the V1 time-stamp trap field
        // When Index = 2,
        //  Oid   = v2snmpTrapOid
        //  value = can be one of the following Oids:
        //          v2coldStartOid
        //          v2warmStartOid
        //          v2linkDownOid
        //          v2linkUpOid
        //          v2authenFailureOid
        //          v2egpNeighborLossOid
        //          v2snmpTrapEnterpriseOid+0+specific_trap
```

```
      // When Index = VBCount, (the last Oid in the v2 trap)
      //  Oid   = v2snmpTrapEnterpriseOid
      //  Value = enterprise specific Oid from V1 trap

     // Get a particular variable from the Varbindlist
     // using the given Index.

      RetStatus = SnmpGetVb (
          hVBL,              // Input Varbindlist Handle
          Index,             // Index to a variable
          &Name,             // Output name of the variable
          &Value);           // Output value of the variable
      // Error condition checking for RetStatus performs here

      SnmpOidToStr (&Name, 100, (LPSTR)NameBuffer);
      SnmpFreeDescriptor (SNMP_SYNTAX_OID, &Name);

      switch (Value.syntax) {

            case SNMP_SYNTAX_INT :
            _ltoa ((long)Value.value.sNumber, ValueBuffer, 10);
            break;

            case SNMP_SYNTAX_UINT32 :
            case SNMP_SYNTAX_CNTR32 :
            case SNMP_SYNTAX_GAUGE32 :
            case SNMP_SYNTAX_TIMETICKS :
            _ltoa ((long)Value.value.uNumber, ValueBuffer, 10);
            break;

            case SNMP_SYNTAX_CNTR64 :
            break;        // Need routine to convert 64-bit number to string here!

            case SNMP_SYNTAX_OCTETS :
            case SNMP_SYNTAX_BITS :
            case SNMP_SYNTAX_OPAQUE :
            case SNMP_SYNTAX_IPADDR :
            case SNMP_SYNTAX_NSAPADDR :
            _fmemcpy (ValueBuffer, Value.value.string.ptr, (size_t)Value.value.string.len);
            SnmpFreeDescriptor (SNMP_SYNTAX_OCTETS, &Value.value.string);
            break;

            case SNMP_SYNTAX_OID :
            SnmpOidToStr (&Value.value.oid, 256, ValueBuffer);
            SnmpFreeDescriptor (SNMP_SYNTAX_OID, &Value.value.oid);
            break;

      } // switch

      OutputVariable (
            Index,             // Index to a given variable for output
            NameBuffer,        // Trap variable Oid Name Output Buffer
            ValueBuffer);      // Trap variable Value Output Buffer
} //for loop

SnmpFreeEntity (hSrc);
```

```
    SnmpFreeEntity (hDest);
    SnmpFreeContext (hContext);
    SnmpFreeVbl (hVBL);
    SnmpFreePdu (hPDU);

    return SNMPAPI_SUCCESS;
} // TrapProcess
```