A Data Structures, Defines, Routines, Flags, and Code Examples
This appendix contains reference information about structures, device
types, packet types, and values used in SAP, TYPE, MTU, kind, and
packet type fields used in networking calls described in Chapter 7,
“Creating Networking Device Drivers.”

It contains reference information for the following sections:

- Interface Group MIB Objects - Contains MIB II structure definitions
  used by network management applications.
- MAC Types and Protocol Types - Contains a list of devices used by
  the hw_ift structure to indicate the device media type.
- Message Types for DLPI Primitive and Acknowledgement - Contains
  definitions of requests and acknowledgments for HP-UX DLPI device
  dependent primitives.
- MIB Event and Event Record - Contains the definition of the event
  message when the driver is required to send an event as part of a call
to network management code.
- MTU Values - Contains a list of the defined MTU values for current
  HP provided protocols.
- Packet Headers - Contains a list of the packet header structures for
  LAN media used by HP-UX
- Packet Types - Contains a list of the packet type values for inbound
  and outbound packets.
- Protocol Kinds and Values - Contains a list of the defined values for
  the dl_proto_kind field of a HP-UX DLPI bind request.
- SAP Values for IEEE 802.2 LLC Packets - Contains a list of the SAP
  values for IEEE802.2 LLC packets.
- TYPE Values for Ethernet and SNAP protocols - Contains a list of
  TYPE values for Ethernet and SNAP packets.
Interface Group MIB Objects

This section contains MIB II structure definitions used by network management applications.

These structures are described in the subsection “Network Management Support” on page 276, Chapter 7, and are defined in the <h/mib.h> header file.

Standard MIB II

typedef struct {
    int ifIndex;
    char ifDescr[64];
    int ifType;
    int ifMtu;
    gauge ifSpeed;
    unsigned char ifPhysAddress[8];
    int ifAdmin;
    int ifOper;
    TimeTicks ifLastChange;
    counter ifInOctets;
    counter ifInUcastPkts;
    counter ifInNUcastPkts;
    counter ifInDiscards;
    counter ifInErrors;
    counter ifInUnknownProtos;
    counter ifOutOctets;
    counter ifOutUcastPkts;
    counter ifOutNUcastPkts;
    counter ifOutDiscards;
    counter ifOutErrors;
    gauge ifOutQlen;
    int ifSpecific;
} mib_ifEntry;
Extended MIB II for 802.3 and 802.5

/* Types for 802.3 extended MIB */

typedef struct {
    int    dot3StatsIndex;
    counter dot3StatsAlignmentErrors;
    counter dot3StatsFCSErrors;
    counter dot3StatsSingleCollisionFrames;
    counter dot3StatsMultipleCollisionFrames;
    counter dot3StatsSQETestErrors;
    counter dot3StatsDeferredTransmissions;
    counter dot3StatsLateCollisions;
    counter dot3StatsExcessiveCollisions;
    counter dot3StatsInternalMacTransmitErrors;
    counter dot3StatsCarrierSenseErrors;
    counter dot3StatsFrameTooLongs;
    counter dot3StatsInternalMacReceiveErrors;
} mib_Dot3StatsEntry;

typedef struct {
    int    dot3CollIndex;
    int    dot3CollCount;
    counter dot3CollFrequencies;
} mib_Dot3CollEntry;

/* Types for 802.5 extended MIB */

typedef char MACADDRESS[6];

typedef struct {
    int    dot5IfIndex;
    int    dot5Commands;
    int    dot5RingStatus;
    int    dot5RingState;
    int    dot5RingOpenStatus;
    int    dot5RingSpeed;
    MACADDRESS dot5UpStream;
    int    dot5ActMonParticipate;
    MACADDRESS dot5Functional;
} mib_Dot5Entry;

typedef struct {
    int    dot5StatsIfIndex;
    counter dot5StatsLineErrors;
    counter dot5StatsLineErrors;
} mib_Dot5StatsEntry;
counter dot5StatsACErrors;
counter dot5StatsAbortTransErrors;
counter dot5StatsInternalErrors;
counter dot5StatsLostFrameErrors;
counter dot5StatsReceiveCongestions;
counter dot5StatsFrameCopiedErrors;
counter dot5StatsTokenErrors;
counter dot5StatsSoftErrors;
counter dot5StatsHardErrors;
counter dot5StatsSignalLoss;
counter dot5StatsTransmitBeacons;
counter dot5StatsRecoverys;
counter dot5StatsLobeWires;
counter dot5StatsRemoves;
counter dot5StatsSingles;
counter dot5StatsFreqErrors;
} mib_Dot5StatsEntry;
MAC Types and Protocol Types

This section contains a list of device types used by the `hw_ifit` structure (`mac_type` field) to indicate the device media type.

- **DEV_8023** For IEEE 802.3 device.
- **DEV_8025** For IEEE 802.5 device.
- **DEV_Ether** For Ethernet device.
- **DEV_FDDI** For FDDI device.
- **DEV_ATM** For ATM device.
- **DEV_FC** For Fibre Channel device.

The flags (defined in `sio/lan_dlpikrn.h`) listed below are used by the `hw_ifit` structure (`llc_flags`) to indicate the protocol type and encapsulation method.

- **IEEE** For IEEE 802.2 type.
- **SNAP** For SNAP type.
- **ETHERTYPE** For Ethernet type.
- **NOVELL** For Novell packet type.

The `hw_ifit` structure is described in “hw_ifit_t Structure Description and Initialization” on page 216, Chapter 7.
Message Types for DLPI Primitive and Acknowledgment

This section contains definitions of requests and acknowledgments for HP-UX DLPI device dependent primitives.

`/*
 * DL_HP_BIND, DL_HP_UNBIND, and DL_HP_LOOKUP_PROTO
 */
typedef struct {
    u_long dl_proto_kind;
    /* Kind will determine size of sap */
    u_char *dl_sap;  /* sap info. */
    u_long (*dl_proto_func)(); /* Interrupt routine */
    u_long dl_proto_info; /* Read queue pointer */
} dl_hp_proto_t;

`/*
 * DL_HP_PROMISCON, DL_HP_PROMISCOFF
 */
typedef struct {
    u_long dl_level;
    /* physical, SAP level or ALL multicast */
    u_long (*dl_proto_func)(); /* Interrupt Routine */
    u_long dl_proto_info; /* Read queue pointer */
} dl_hp_promiscon_t;

`/*
 * DL_HP_ENABMULTI_ADDR, DL_HP_DISABMULTI_ADDR,
 * DL_HP_SET_PHYS_ADDR, DL_HP_GET_PHYS_ADDR, and
 * DL_HP_GET_STATISTICS.
 */
typedef struct {
    u_long dl_data_len;
    u_char *dl_data;
} dl_hp_data_t;

Supported DLPI primitives are summarized in “Summary of DLPI Primitives and IOCTLs” on page 256, Chapter 7.
MIB Event and Event Record

This section defines the event message that the driver is required to send an event as part of a call to network management code.

```c
struct evrec {
    struct event ev;
    struct evrec *evnext;
}

struct event {
    timeval time; /* timestamp */
    int code; /* event code */
    int len; /* byte count of data in info */
    char info[MAXEVINFO]; /* event specific info */
}
```
MTU Values

This section lists the defined MTU values (Maximum Transmission Unit without header) for the current HP provided protocols.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETHERMTU</td>
<td>1500 bytes, max Ethernet packet size.</td>
</tr>
<tr>
<td>IEEE8023_MTU</td>
<td>497 bytes, max IEEE 802.3 packet size.</td>
</tr>
<tr>
<td>SNAP8023_MTU</td>
<td>492 bytes, max SNAP 802.3 packet size.</td>
</tr>
<tr>
<td>IEEE8025_4_MTU</td>
<td>4170 bytes, max packet size for 4M bit Token Ring.</td>
</tr>
<tr>
<td>SNAP8025_4_MTU</td>
<td>4170 bytes, max SNAP packet size for 4M bit Token Ring.</td>
</tr>
<tr>
<td>IEEE8025_16_MTU</td>
<td>4170 bytes, max packet size for 16M bit Token Ring.</td>
</tr>
<tr>
<td>SNAP8025_16_MTU</td>
<td>4170 bytes, max SNAP packet size for 16M bit Token Ring.</td>
</tr>
<tr>
<td>FDDI_MTU</td>
<td>4352 bytes, max SNAP packet size for FDDI.</td>
</tr>
</tbody>
</table>

These values are defined in the netinet/if_ether.h, netinet/if_ieee.h, and sio/fddio.h files.
Packet Headers

This section lists the packet header structures for LAN media used in HP-UX.

```c
struct ether_hdr {    
    u_char destaddr[6];  
    /* Ethernet destination address */ 
    u_char sourceaddr[6];  
    /* Ethernet source address */ 
    u_short type; /* Ethernet type value */ 
};

struct ieee8023_hdr {    
    u_char destaddr[6];  
    /* IEEE 802.3 destination address */ 
    u_char sourceaddr[6];  
    /* IEEE 802.3 source address */ 
    u_short length; /* byte count of packet length */ 
    u_char dsap; /* dsap value */ 
    u_char ssap; /* ssap value */ 
    u_char ctrl; /* ctrl value */ 
};

struct snap8023_hdr {    
    u_char destaddr[6];  
    /* IEEE 802.3 destination address */ 
    u_char sourceaddr[6];  
    /* IEEE 802.3 source address */ 
    u_short length; /* byte count of packet length */ 
    u_char dsap; /* dsap value */ 
    u_char ssap; /* ssap value */ 
    u_char ctrl; /* ctrl value */ 
    u_char hdr_fill[3]; /* padding for alignment */ 
    u_short type; /* type value */ 
};

struct ieee8024_hdr {    
    u_char frame_ctrl; /* frame control field */ 
    u_char destaddr[6];  
    /* IEEE 802.4 destination address */ 
    u_char sourceaddr[6];  
    /* IEEE 802.4 source address */ 
    u_char dsap; /* dsap value */ 
    u_char ssap; /* ssap value */ 
    u_char ctrl; /* ctrl value */ 
};
```
Data Structures, Defines, Routines, Flags, and Code Examples

Packet Headers

```c
struct snap8024_hdr {
    u_char frame_ctrl; /* frame control field */
    u_char destaddr[6];
    /* IEEE 802.4 destination address */
    u_char sourceaddr[6];
    /* IEEE 802.4 source address */
    u_char dsap;       /* dsap value */
    u_char ssap;       /* ssap value */
    u_char ctrl;       /* ctrl value */
    u_char hdr_fill[3]; /* padding for alignment */
    u_char type[2];    /* type value */
};

struct ieee8025_sr_hdr {
    u_char access_ctl; /* access control field */
    u_char frame_ctrl; /* frame control field */
    u_char destaddr[6];
    /* IEEE 802.5 destination address */
    u_char sourceaddr[6];
    /* IEEE 802.5 source address */
    u_char rif[18]; /* IEEE 802.5 source routing information */
    u_char dsap;       /* dsap value */
    u_char ssap;       /* ssap value */
    u_char ctrl;       /* ctrl value */
};

struct snap8025_sr_hdr {
    u_char access_ctl; /* access control field */
    u_char frame_ctrl; /* frame control field */
    u_char destaddr[6];
    /* IEEE 802.5 destination address */
    u_char sourceaddr[6];
    /* IEEE 802.5 source address */
    u_char rif[18]; /* IEEE 802.5 source routing information */
    u_char dsap;       /* dsap value */
    u_char ssap;       /* ssap value */
    u_char ctrl;       /* ctrl value */
    u_char orgid[3];   /* organization ID */
    u_short type;      /* type value */
};

struct fddi_hdr {
    u_char pad[3];     /* pad characters */
    u_char fc;         /* frame control field */
    u_char destaddr[6];
    /* IEEE 802.5 destination address */
    u_char sourceaddr[6];
};
```
Packet Headers

```c
/* IEEE 802.5 source address */
   u_char dsap;    /* dsap value */
   u_char ssap;    /* ssap value */
   u_char ctrl;    /* ctrl value */
}
struct snapfddi_hdr {
   u_char pad[3];  /* pad characters */
   u_char fc;      /* frame control field */
   u_char destaddr[6];
} /* IEEE 802.5 destination address */
struct sourceaddr[6];
/* IEEE 802.5 source address */
   u_char dsap;    /* dsap value */
   u_char ssap;    /* ssap value */
   u_char ctrl;    /* ctrl value */
   u_char orgid[3];
/* organization ID = 00, 00, 00 */
   u_short type;
/* type value; IP = 0x800, ARP = 0x806 */
}
struct snapfddi_hdr_info {
   u_char destaddr[6]; /* FDDI destination address */
   u_short type;      /* type value */
}
struct ieee8022_hdr {
   u_char dsap;    /* dsap value */
   u_char ssap;    /* ssap value */
   u_char ctrl;    /* ctrl value */
}
struct snap8022_hdr {
   u_char dsap;    /* dsap value */
   u_char ssap;    /* ssap value */
   u_char ctrl;    /* ctrl value */
   u_char orgid[3];/* organization ID */
   u_short type;   /* type value */
}

These structures are contained in the netinet/if_ether.h, netinet/if_ieee.h, and sio/fddiio.h header files.
Packet Types

This section lists the packet type values for inbound and outbound packets.

ETHER_PKT Ethernet packet.
SNAP8023_PKT SNAP packet over IEEE 802.3 media.
IEEE8023_PKT IEEE 802.3 packet.
SNAP8025_PKT SNAP packet over IEEE 802.5 media.
IEEE8025_PKT IEEE 802.5 packet.
SNAPFDDI_PKT SNAP packet over FDDI media.
SNAPFDDI_LLA_PKT SNAP (for DLPI) packet over FDDI media.
FDDI_UI_PKT Native FDDI packet.
FDDI_LLA_PKT Native FDDI (for DLPI) packet.

These packets are defined in the netinet/if_ether.h file.
Protocol Kinds and Values

This section lists the defined values for the `dl_proto_kind` field of HP-UX DLPI bind request.

```c
enum protocol_kinds {LAN_SAP, LAN_TYPE, LAN_CANON, LAN_SNAP, LAN_SNAP_EXT}
```

- **LAN_SAP** For IEEE 802.[2/3] packet with SAP values as part of protocol format.
- **LAN_TYPE** For Ethernet packet with TYPE value as part of protocol format.
- **LAN_SNAP** For SNAP type of protocol format.
- **LAN_SNAP_EXT** For SNAP extension type protocol format.

These kinds are enumerated in the `sio/lan_dlpikrn.h` header described in “STREAMS DLPI and Network Driver Overview” on page 242, chapter 7.
SAP Values for IEEE 802.2 LLC Packets

This section lists the SAP values, defined in `netinet/if_EISA.h`, for IEEE 802.2 LLC packets.

- `IEEESAP_IP` 0x06, for IP protocol
- `IEEESAP_SNAP` 0xAA, for SNAP protocol
TYPE Values for Ethernet and SNAP Protocols

This section lists the TYPE values, defined in netint/if_ether.h, for Ethernet and SNAP packets.

ETHERTYPE_IP 0x0800, for IP protocol

ETHERTYPE_ARP 0x0806, for ARP protocol