Interface Management API Implementation Agreement (PDH Interfaces)

Revision 3.0

Editor: Tim Shanley, TranSwitch, tim.shanley@transwitch.com

The words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", “MAY", and "OPTIONAL" in the remainder of this document are to be interpreted as described in the NPF Software API Conventions Implementation Agreement revision 2.0.

For additional information contact:
The Network Processing Forum, 39355 California Street,
Suite 307, Fremont, CA 94538
+1 510 608-5990 phone info@npforum.org

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The following data and functions definitions are proposed to NP Forum for inclusion in the next revision of the Interface Management API.

The NP Forum ATM Task Group will be defining a set new API to manage ATM connections; as a result of that work, the Interface Mangement API will need to support the specific interface types commonly used to transport ATM. These are SONET, SDH, and Plesiochronous Digital Hierarch (PDH) interfaces, such as DS0/DS1/DS2/DS3, T1/E1, T3/E3, interfaces.

The use of data structures relationships for abstracting and representing of the relationships between the sub-layers of a particular protocol has a particular importance in the case of **Plesiochronous Digital Hierarchy (PDH)**, **Synchronous Optical Network (SONET)** and **Synchronous Digital Hierarchy (SDH)** networks (npf2004.115). These are underlying networks widely used in both voice and data communications and constitute a major building block for the Metropolitan and Wide Area Networks.

Note: The name Plesiochronous Digital Hierarchy (PDH) stems from the timing of signals across the network, which is plesiochronous, which means almost but not precisely synchronous.

1.1 PDH Digital Hierarchy

In a simplified description, in a PDH Digital Hierarchy, there are multiple channels carrying digitized voice or data. The channel is the lowest or smallest bandwidth path. The basic channel is DS0, which can carry 64Kb/sec.

DS0s can be grouped - 24 DS0s – into a DS1, which is equivalent to T1 (24 DS0 channels). A DS1 frame has 193 bits, carrying 24 8 bit DS0 channels, at a sampling rate of 8Khz. This results in a bandwidth capacity of 1.544 Mb/s.

DS1s can be grouped – 4 DS1 – into a DS2, which is equivalent to T2 (96 DS0 channels). A DS2 frame contains 4 subframes. Each subframe consists of 6 blocks of 49 bits each. The DS2 frame contains interleaved bits of the DS1 frames – a total of 1551 DS1 bits. This results in a bandwidth of 6.176Mb/s.

DS2, or DS1s can be grouped – 7 DS2s, or 28 DS1s, or 672 DS0s – into a DS3, which is equivalent to T3 (672 DS0 channels). The DS3 frame consists of 7 subframes, each consisting of 8 blocks and each block containing 85 bits. This results in a bandwidth of 44.184Mb/s.

The sharing of this bandwidth by channels, in simplified description, is like sharing the circuit in rotation, with each DS0 having its own assigned time slot to use or not as the case may be. As each channel is always found in the same place, in the frame, no address is needed to extract (demultiplex) that channel at the destination.

The PDH Hierarchy, or the grouping of DS0 channels into DS1s, DS2s, DS3s, can be abstracted and represented by a corresponding Interface Management data structure hierarchy of interfaces in parent-child relationships.

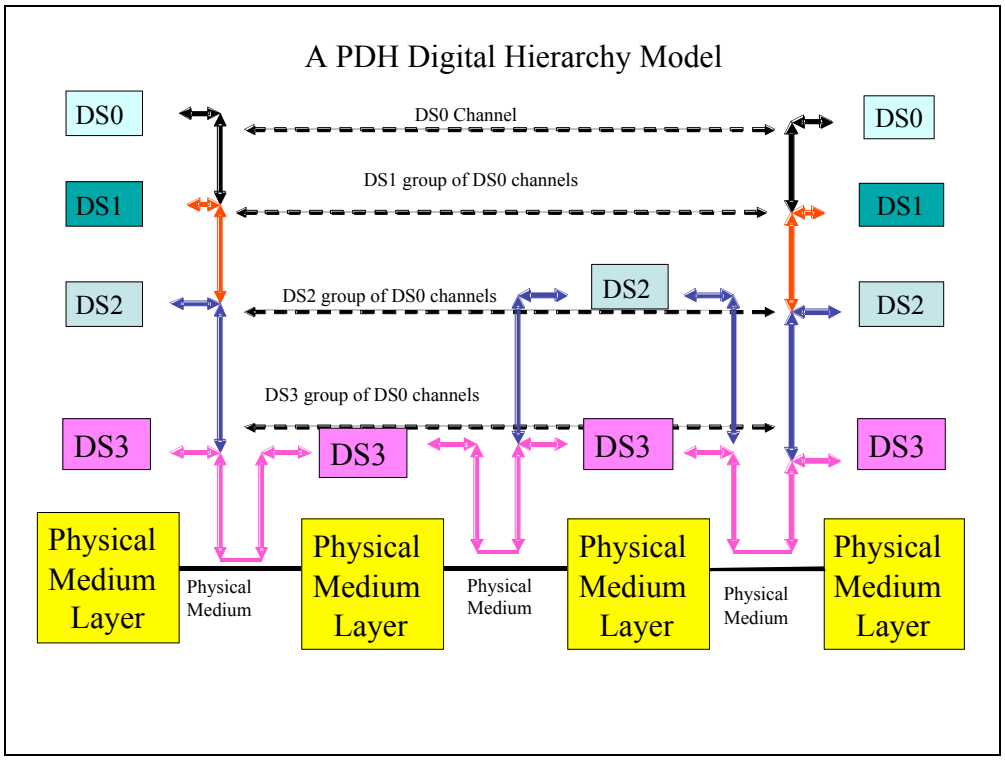

Figure 1-1 PDH Hierarchy

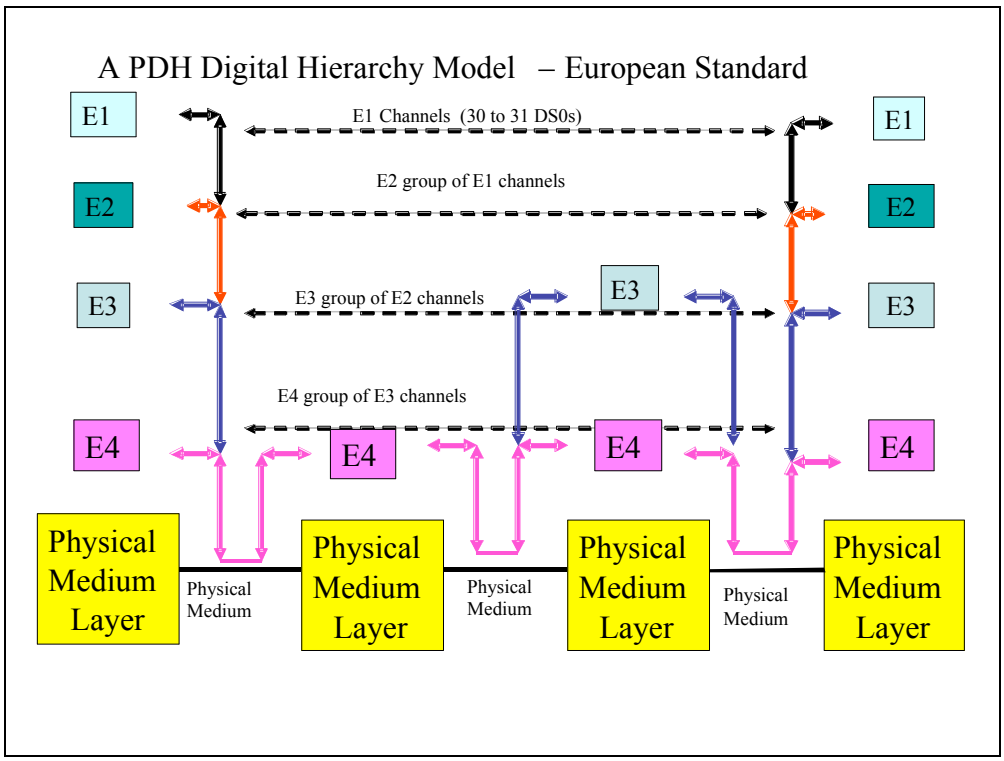

Figure 1-2 A PDH Digital Hierarchy Model – European Standard

An illustration of the types of PHD interfaces is in Figure 1-4

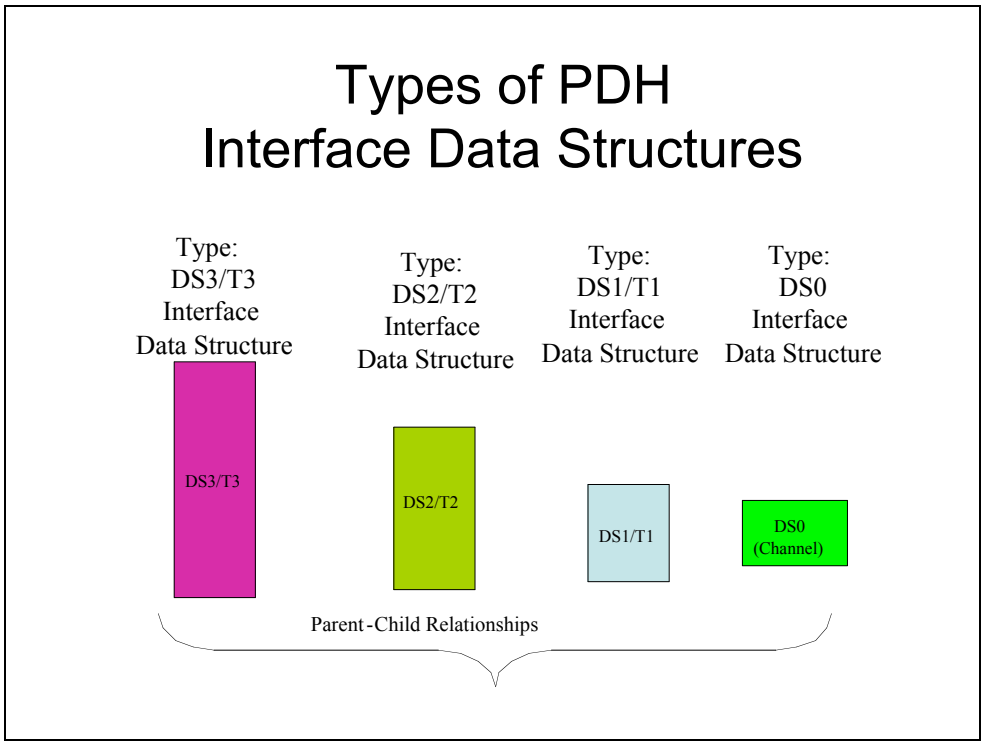

Figure 1-3 Types of PDH Interfaces – American Standard

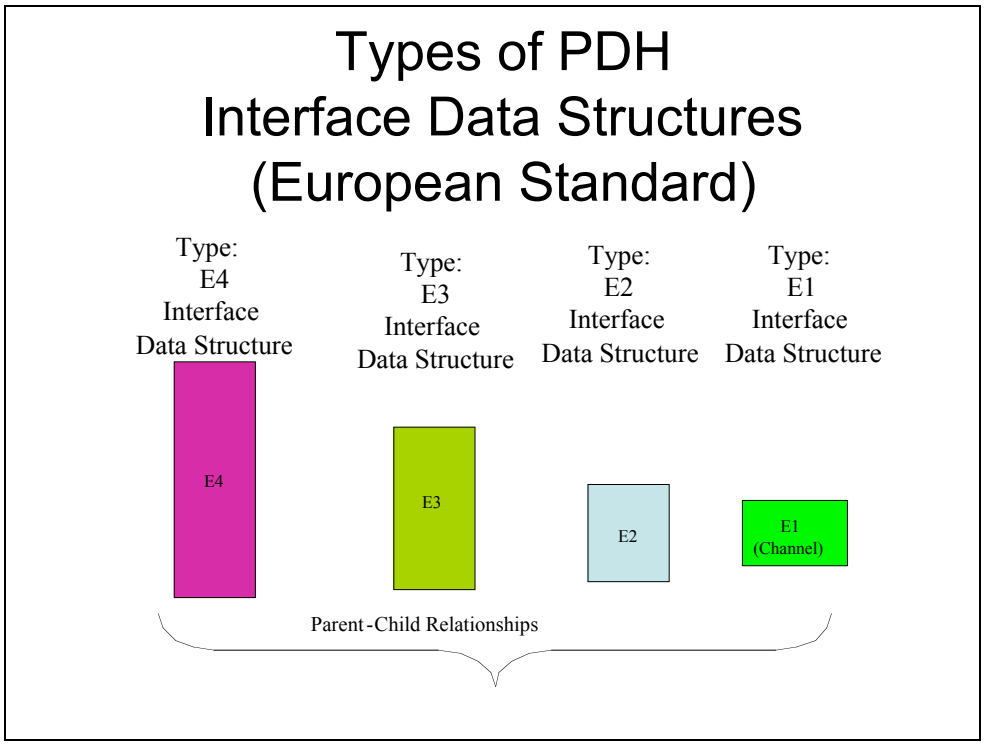

Figure 1-4 PDH Interface Types – European Standard

An illustration of the PDH interface parent-child relationships is illustrated in Figure 1-5

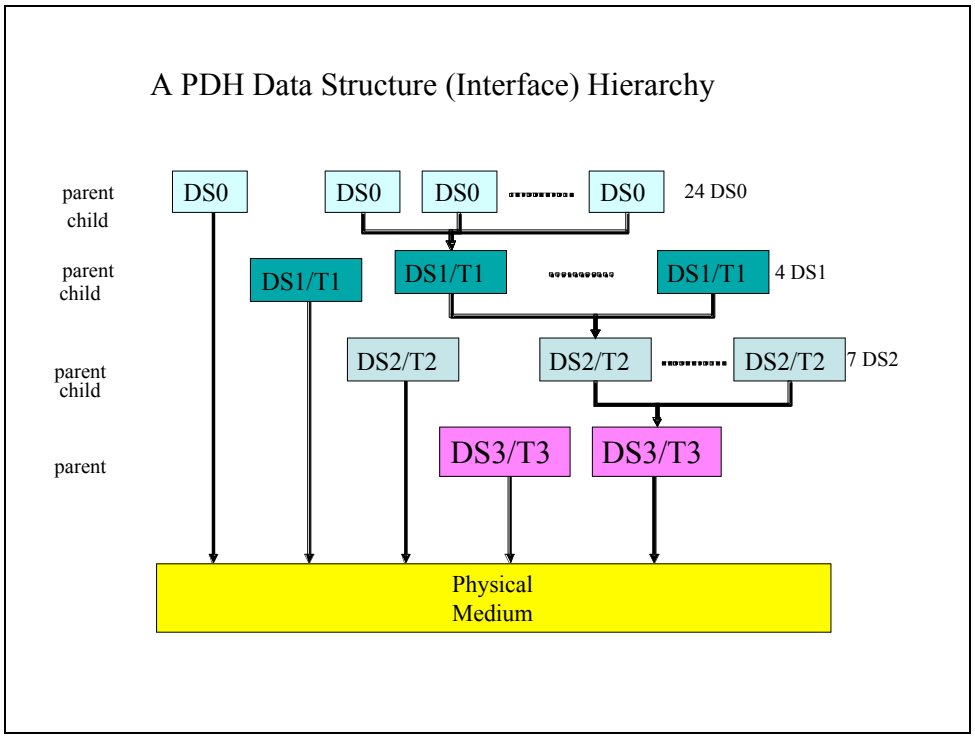

Figure 1-5 PDH Interface Parent-Child Relationships

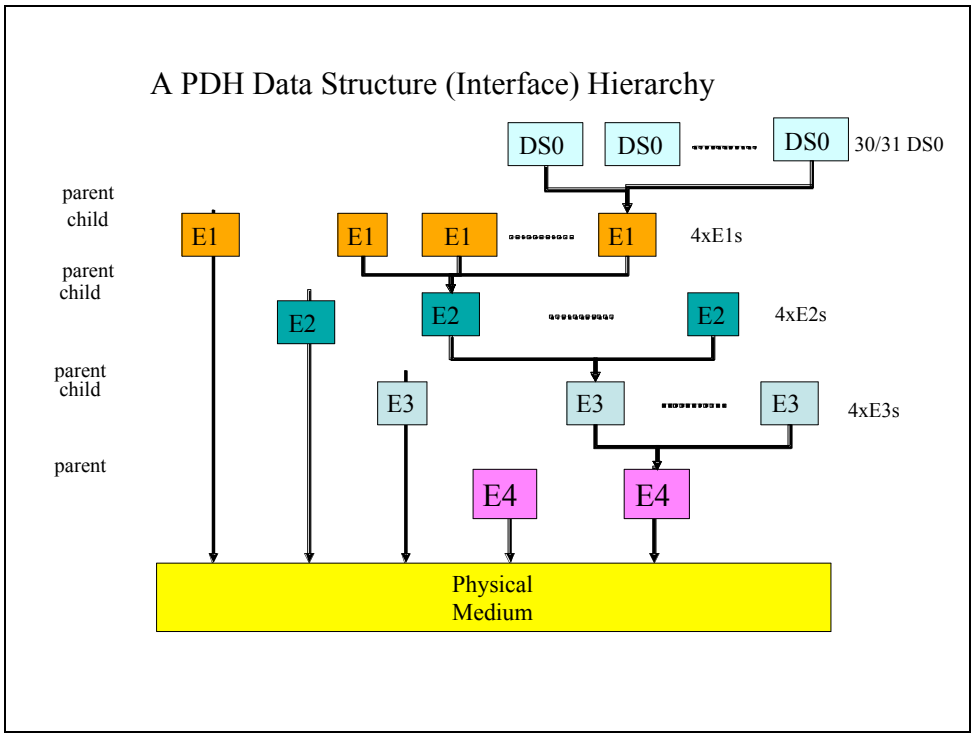

Figure 1-6 A PDH Data Structure Hierarchy – European Standard

1.2 PDH and SDH or SONET Digital Hierarchies

The PDH, SDH, and SONET digital hierarchies are very tightly related. The PDH hierarchy provides tributaries into the SDH or SONET hierarchies. In other words, PDH signals and frame formats are mapped into SDH or SONET signals and frame formats. This tight relationship is captured in the parent-child relationships between PDH and SDH or SONET interfaces.

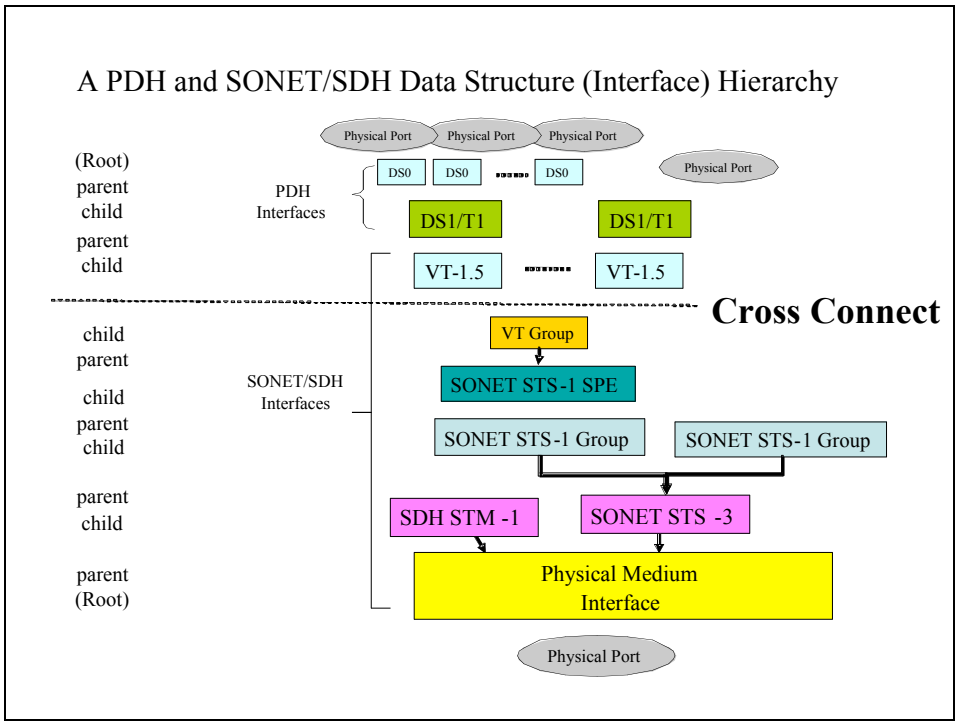

Figure 1-7 PDH and SONET/SDH interface relationships

1.3 PDH Parent-Child Interfaces in a Protocol Stack

The use of data structures relationships for abstracting and representing of the protocol layering in a protocol stack is important. Such interface relationships establish an interface tree structure, in which the closest interface to the physical port is the Root, and the highest interface in the protocol stack is the highest branch in the tree. Layers in the tree are in parent-child relationship. An illustration of this is Figure 1-8

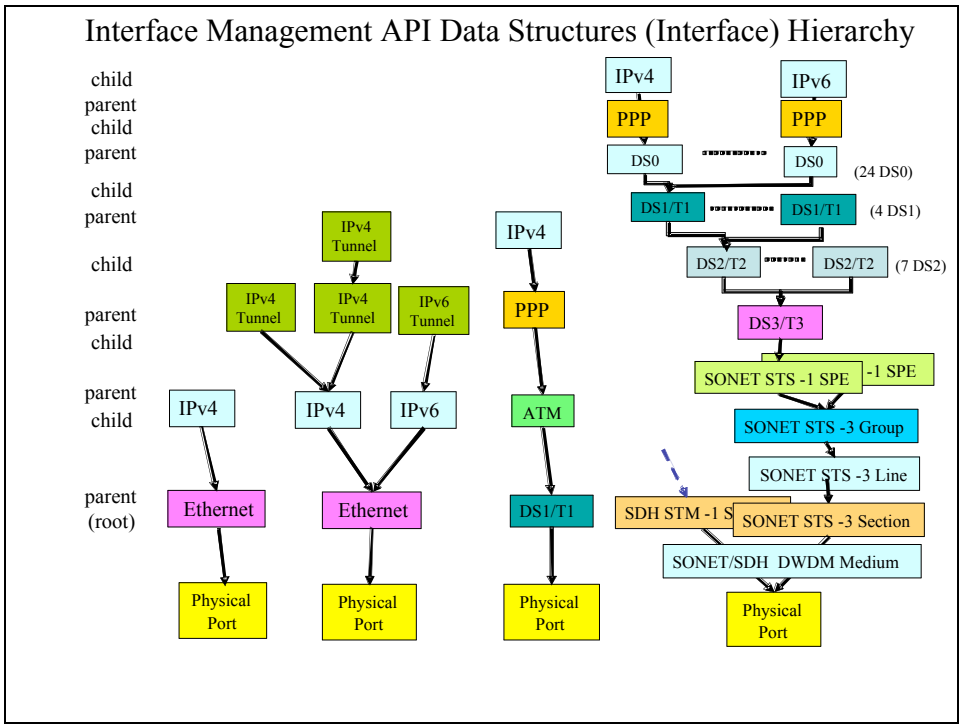

Figure 1-8 Interface Data Structures Parent-Child Relationships

1.4 PDH API Functional Areas

There are several areas that the PDH Interface Management API extensions are going to address:

1.4.1 PDH Network Element General Configuration

Configuring a PDH network element (NE) consists of configuring the PDH parameters of the components so that the NE materialize a certain PDH network multilayered hierarchy.

1.4.2 PDH Fault Management - Alarm/Status Monitoring

Fault Management (FM) - Alarm/Status Monitoring - is a process that tracks failure events to contribute to understanding the overall transmission performance of a PDH network element, and/or the components of a PDH Network Element (NE) (see Figure 1-9).

This information conveyed via alarm/status monitoring consists of a set of binary data, known as indications that are maintained by the NE, or its components. The NE or its components sets and clears indications according to well-defined standard criteria based on the occurrence and duration of specific events. Some events lead immediately to indications, while others must persist for a specified amount of time prior to setting of an indication.

1.4.3 PDH Performance Monitoring

Performance Monitoring (PM) is a process of continuously collecting, analyzing, and reporting performance data associated with a network element, or transmission entity. It refers to the set of functions and capabilities necessary to collect, store, threshold, and report performance data (see Figure 1-9).

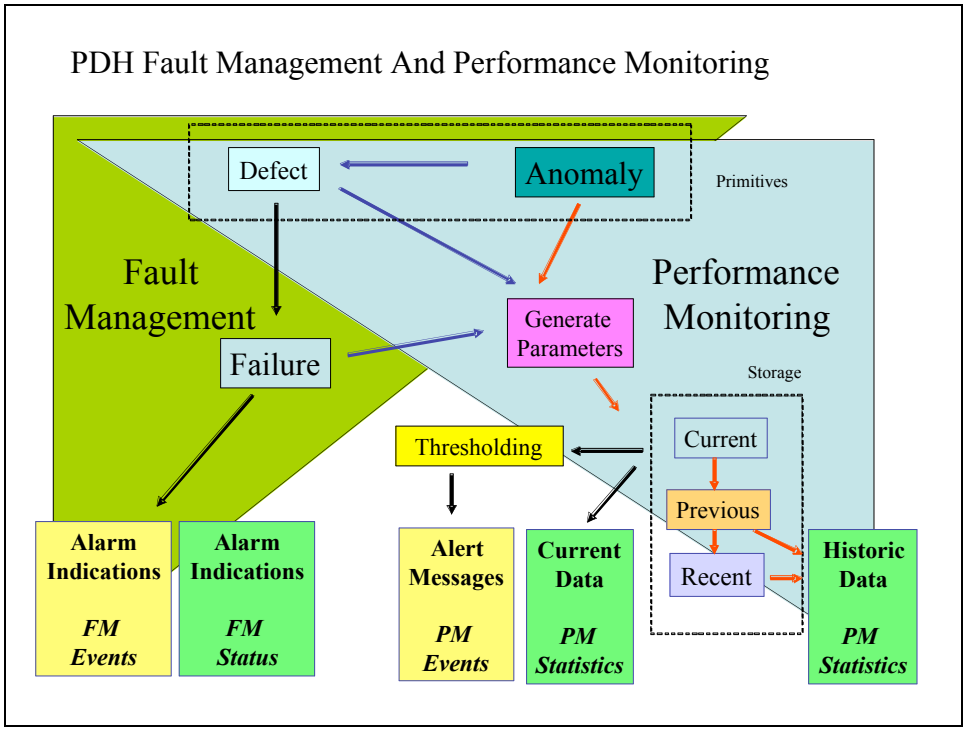

Figure 1-9 Fault Management and Performance Monitoring

2 PDH NPF Interface Management Data Types

This section describes the PDH Interface Management data structures.

2.1 PDH NPF Interface Data Structures

2.1.1 PDH Interface Configuration Data Structures

2.1.1.1 PDH Interface Type Code

The PDH interfaces are part of the PDH interface type. For supporting the PDH interface, the PDH interface type must exist in the Core Interface Management API interface structure.

```
#define     NPF_IF_TYPE_PDH   11          /* PDH interface */
```

2.1.1.1.1 PDH Interface Types

The following PDH interface types are defined:

```
  typedef enum    {

    NPF_IF_PDH_TYPE_DS0        = 1,        /* US Standard */
    NPF_IF_PDH_TYPE_DS0_BUNDLE = 2,        /* US Standard */
    NPF_IF_PDH_TYPE_DS1        = 3,        /* US standard */
    NPF_IF_PDH_TYPE_DS2        = 4,        /* US standard */
    NPF_IF_PDH_TYPE_DS3        = 5,        /* US standard */
/**/
    NPF_IF_PDH_TYPE_T1         = 6,        /* US standard */
    NPF_IF_PDH_TYPE_T2         = 7,        /* US standard */
    NPF_IF_PDH_TYPE_T3         = 8,        /* US standard */
/**/
    NPF_IF_PDH_TYPE_E1         = 9,        /* European standard */
    NPF_IF_PDH_TYPE_E2         = 10,       /* European standard */
    NPF_IF_PDH_TYPE_E3         = 11,       /* European standard */
    NPF_IF_PDH_TYPE_E4         = 12,       /* European standard */
/**/
    NPF_IF_PDH_TYPE_J1         = 13,       /* Japanese standard 1 */
    NPF_IF_PDH_TYPE_J2         = 14        /* Japanese standard 2 */

  } NPF_IfPDH_Type_t;
```

2.1.1.2 PDH Interface Attributes

A forward reference to the PDH NPF Interface Attributes data structure must exist in the **`NPF_IfGeneric_t`** structure in **`npf_if_core.h`** if PDH interfaces are supported. So, before the declaration of **`NPF_IfGeneric_t`**, the following must appear:

```
typedef struct NPF_IfPDH NPF_IfPDH_t;
```

The following must also appear inside the union within the **`NPF_IfGeneric_t`** structure:

```
        NPF_IfPDH_t *PDH_Attr; /* PDH interface attributes */
```

The following data structure contains configuration parameters for the PDH interface type. It is used when setting, or querying PDH interface attributes.

```
/*
**  PDH Interface Attributes
*/
```

```
struct NPF_IfPDH {

      NPF_uint32_t      port;       /* Port number */
      NPF_IfPDH_Type_t  PDHType;    /* PDH Interface type */

      NPF_IfPDH_LineSpeed_t   LineSpeed;
      NPF_IfPDH_LineType_t    LineType;
      NPF_IfPDH_LineCoding_t  LineCode;
      NPF_IfPDH_SendCode_t    SendCode;
      NPF_IfPDH_Loopback_t    Loopback;
/**/
      NPF_IfPDH_SignalMode_t  SignalMode;
      NPF_IfPDH_TxClSrc_t     XMTClockSource;
      NPF_IfPDH_FDLink_t      FacilDataLink;
/**/
      NPF_uint32_t      InvalidIntervals[];
      NPF_uint32_t      LineLength;
      NPF_uint32_t      LineStatusLastChange[];
/**/
      NPF_uint32_t      TimeElapsed;
      NPF_uint32_t      ValidIntervals;
/**/
      NPF_IfPDH_LineStatus_t        LineStatus;
      NPF_IfPDH_LoopbackStatus_t    LoopbackStatus;
/**/


          } ;
```

2.1.1.2.1 PDH Line Speed Interface Attribute

```
/*
** Line Speed
*/

typedef enum {

      NPF_IfPDH_DS1 = 1,   /* DS1 – 1.544 Mbit/sec */
      NPF_IfPDH_E1 = 2,    /* E1  - 2.048 Mbit/sec */
      NPF_IfPDH_DS2 = 3,   /* DS2 – 6.312 Mbit/sec */
      NPF_IfPDH_E2 = 4,    /* E2  - 8.448 Mbit/sec */
      NPF_IfPDH_DS3 = 5,   /* DS3 – 44.736 Mbit/sec */
      NPF_IfPDH_E3 = 6,    /* E3 – 34.368 Mbit/sec */
      NPF_IfPDH_E4 = 7     /* E3 – 139.264 Mbit/sec */

      } NPF_IfPDH_LineSpeed_t
```

2.1.1.2.2 PDH Line Type Interface Attribute

```
/*
** Line Type
*/
typedef enum {

NPF_IF_PDH_LINE_TYPE_OtherType  = 0,
            /* Type other */
```

```
NPF_IF_PDH_LINE_TYPE_DS1_ESF = 1,
            /*  Extended SuperFrame DS1 (T1.107) */
NPF_IF_PDH_LINE_TYPE_DS1_D4 = 2,
            /*  AT&T D4 format DS1 (T1.107) */
NPF_IF_PDH_LINE_TYPE_E1 = 3,
            /* ITU-T Recommendation G.704 */
NPF_IF_PDH_LINE_TYPE_E1_CRC = 4,
            /* ITU-T Recommendation G.704 */
NPF_IF_PDH_LINE_TYPE_E1_MF = 5,
            /* G.704 with TS16 multi-framing enabled */
NPF_IF_PDH_LINE_TYPE_E1_CRCMF = 6,
            /* G.704 with TS16 multi-framing enabled */
NPF_IF_PDH_LINE_TYPE_DS1_Unframed = 7,
            /* DS1 with no framing */
NPF_IF_PDH_LINE_TYPE_E1_Unframed = 8,
            /* E1 with no framing */
      NPF_IF_PDH_LINE_TYPE_DS2_M12 = 9,
                  /* DS2 frame format (T.107) */
NPF_IF_PDH_LINE_TYPE_E2 = 10,
            /* E2 Frame format (G.704) */
NPF_IF_PDH_LINE_TYPE_DS3_M23 = 11,
            /* ANSI T1.107-1988 */
NPF_IF_PDH_LINE_TYPE_DS3_SYNTRAN = 12,
            /* ANSI T1.107-1988 */
NPF_IF_PDH_LINE_TYPE_DS3_CbitParity = 13,
            /* ANSI T1.107a-1990 */
NPF_IF_PDH_LINE_TYPE_DS3_ClearChannel = 14,
/* ANSI T1.102-1987 */
NPF_IF_PDH_LINE_TYPE_E3_Framed = 15,
            /* CCITT G.751*/
NPF_IF_PDH_LINE_TYPE_E3_Plcp = 16,
            /* ETSI T/NA (1998) */
NPF_IF_PDH_LINE_TYPE_E3_G832 = 17,
            /* */
NPF_IF_PDH_LINE_TYPE_E4 = 18,
            /* */
NPF_IF_PDH_LINE_TYPE_E4_G832 = 19
            /* */


} NPF_IfPDH_LineType_t;
```

2.1.1.2.3 PDH Line Coding Interface Attribute

```
/*
** Line Coding
*/
typedef enum {
      NPF_IF_PDH_LineCoding_Other = 0,
      NPF_IF_PDH_LineCoding_DS1_JBZS  = 1,
      NPF_IF_PDH_LineCoding_DS1_B8ZS  = 2,
      NPF_IF_PDH_LineCoding_DS1_HDB3  = 3,
      NPF_IF_PDH_LineCoding_DS1_ZBTSI  = 4,
      NPF_IF_PDH_LineCoding_DS1_AMI = 5,
      NPF_IF_PDH_LineCoding_DS1_B6ZS = 6,
/* DS3 Lien Coding */
```

```
      NPF_IF_PDH_LineCoding_DS3_B3ZS = 8,
      NPF_IF_PDH_LineCoding_E3_HDB3 = 9,
      NPF_IF_PDH_LineCoding_E4_CMI = 10


}NPF_IfPDH_LineCoding_t
```

2.1.1.2.4 PDH Send Code Interface Attribute

```
/*
** Send Code
*/

typedef enum {

         NPF_IF_PDH_SendNoCode = 1,
         NPF_IF_PDH_SendLineCode = 2,
         NPF_IF_PDH_SendPayloadCode = 3,
         NPF_IF_PDH_SendResetCode = 4,

/* */

         NPF_IF_PDH_DS1_SendQRS = 5,
         NPF_IF_PDH_DS1_Send511Pattern = 6,
         NPF_IF_PDH_DS1_Send3in24Pattern = 7,
         NPF_IF_PDH_DS1_SendOtherTestPattern = 8,

/* */

         NPF_IF_PDH_DS3_SendDS1LoopCode = 13,
         NPF_IF_PDH_DS3_SendTestPattern = 14

      } NPF_IfPDH_SendCode_t;
```

2.1.1.2.5 PDH Loopback Interface Attribute

```
/*
** Loopback
*/

typedef enum {

      NPF_IF_LineLoop  = 1,
      NPF_IF_PayloadLoop = 2,
      NPF_IF_Inward = 3,
      NPF_IF_Dual = 4,
      NPF_IF_NoLoopback = 5,
      NPF_IF_OtherLoop = 6

      } NPF_IfPDH_Loopback_t;
```

2.1.1.2.6 PDH Signal Mode Interface Attribute

```
/*
** SignalMode
*/
typedef enum {

      NPF_IF_PDH_None = 1,
      NPF_IF_PDH_RobbedBit = 2,
      NPF_IF_PDH_BitOriented = 3,
      NPF_IF_PDH_MessageOriented = 4,
      NPF_IF_PDH_SignalMode_Other = 5

      } NPF_IfPDH_SignalMode_t;
```

2.1.1.2.7 PDH Transmit Clock Source Interface Attribute

```
/*
** Transmit Clock Source
*/
typedef enum{

      NPF_IF_PDH_loopTiming = 1,
      NPF_IF_PDH_localTiming = 2,
      NPF_IF_PDH_throughtiming = 3

      } NPF_IfPDH_TxClSrc_t;
```

2.1.1.2.8 PDH Facilities Data Link Interface Attribute

```
/*
**  Facilities Data Link
*/
typedef enum{

      NPF_IF_PDH_Other = 1,
      NPF_IF_PDH_Dsx1ANSIT1403 = 2,
      NPF_IF_PDH_ATT54016 = 3,
      NPF_IF_PDH_FDLNone = 4

      } NPF_IfPDH_FDLink_t;
```

2.1.2 PDH Fault Management Data structures

2.1.2.1 PDH Interface Fault Management Attributes

2.1.2.1.1 PDH Line Status Interface Attribute

```
/*
** Line Status
** PDH Line Specific Status bits
*/
typedef enum {
         NPF_IF_PDH_NoAlarm  = 1,
                              /* No alarm present */
```

```
        NPF_IF_PDH_DS1_RcvFarEndLOF = 2,
                              /* Far end LOF (a.k.a., Yellow Alarm) */
        NPF_IF_PDH_DS1_XmtFarEndLOF = 3,
                              /* Near end sending LOF Indication */
        NPF_IF_PDH_RcvAIS = 4,
                                   /* Far end sending AIS */
        NPF_IF_PDH_XmtAIS = 5,
                                     /* Near end sending AIS */
        NPF_IF_PDH_LOF = 6,
                                 /* Near end LOF (a.k.a., Red Alarm) */
        NPF_IF_PDH_LOS = 7,
                                   /* Near end Loss Of Signal */
        NPF_IF_PDH_LoopbackState = 8,
                                         /* Near end is looped */
        NPF_IF_PDH_DS1_T16AIS = 9,
                                       /* E1 TS16 AIS */
        NPF_IF_PDH_DS1_RcvFarEndLOMF = 10,
                                   /* Far End Sending TS16 LOMF */
        NPF_IF_PDH_DS1_XmtFarEndLOMF = 11,
                                   /* Near End Sending TS16 LOMF */
        NPF_IF_PDH_RcvTestCode   = 12,
                              /*  Near End detects a test code */
        NPF_IF_PDH_OtherFailure  = 13,
                              /* any line status not defined here */
        NPF_IF_PDH_UnavailSigState = 14,
                           /* Near End in Unavailable Signal State */
        NPF_IF_PDH_NetEquipOOS = 15,
                              /* Carrier Equipment Out of Service */
        NPF_IF_PDH_DS2_RcvPayloadAIS = 16,
                                   /* DS2 Payload AIS */
        NPF_IF_PDH_DS2_PerfThreshold = 17,
                              /* DS2 Performance Threshold Exceeded */
        NPF_IF_PDH_DS3_RcvRAIFailure  = 18,
                         /*  Receiving Yellow/RemoteAlarm Indication */
        NPF_IF_PDH_DS3_XmitRAIAlarm    = 19
                         /* Transmit Yellow/RemoteAlarm Indication */
    } NPF_IfPDH_LineStatusBits_t;



struct NPF_IfPDH_LineStatus{
       NPF_uint16_t    n_Statusbits;
       NPF_IfPDH_LineStatusBits_t  *Statusvalue;
};
```

2.1.2.1.2 PDH Interface Loopback Status

```
/*
** Loopback Status
*/
typedef enum {

      NPF_IF_PDH_NoLoopback = 1,
      NPF_IF_PDH_NearEndPayloadLoopback = 2,
      NPF_IF_PDH_NearEndLineLoopback = 3,
      NPF_IF_PDH_NearEndOtherLoopback = 4,
      NPF_IF_PDH_NearEndInwardLoopback = 5,
      NPF_IF_PDH_FarEndPayloadLoopback = 6,
```

```
      NPF_IF_PDH_FarEndLineLoopback = 7

      } NPF_IfPDH_LoopbackStatusBits_t ;

struct NPF_IfPDH_LoopbackStatus{
             NPF_uint16_t    n_bits;
             NPF_IfPDH_LoopbackStatusBits_t  *value;
      };
```

2.1.3 PDH Performance Monitoring Data Structures

The PDH statistics returned by the invocation of the Performance Monitoring Function call are gathered in a union of structures. Based on the interface type, the appropriate member structure of the union is being used:

```
/*
** PDH Statistics
*/

union NPF_IfPDH_Statistics {

   NPF_IfPDH_DS1_DS2_Stats_t  DS1_DS2Stats;
   NPF_IfPDH_DS3_Stats_t  DS3Stats;

   };
```

2.1.3.1 DS1-DS2 Performance Monitoring Statistics structures

DS1-DS2 Performance Monitoring Statistics are gathered on the following:

- Line Errored Seconds (LES)
- Controlled Slip Seconds (CSS)
- Errored Seconds (ES)
- Bursty Errored Seconds (BES)
- Severely Errored Seconds (SES)
- Severely Errored Framing Second (SEFS)
- Degraded Minutes (DM)
- Unavailable Seconds (UAS)

The structure returned by a Query Statistics function calls would contain the following structure:

```
/*
** DS1-DS2 Local statistics
*/

typedef struct {
```

```
    NPF_IfPDH_CurStats_t        CurStats;
    NPF_IfPDH_IntervalStats_t   IntervalStats;
    NPF_IfPDH_CurStats_t        TotalStats;

    } NPF_IfPDH_DS1_DS2_LocalStats_t;
```

2.1.3.1.1 DS1-DS2 Local and Far End Statistics

```
/*
** DS1-DS2 Statistics
*/

typedef struct {

    NPF_IfPDH_DS1_DS2_LocalStats_t  LocalStats;
    NPF_IfPDH_DS1_DS2_LocalStats_t  FarEndStats;

   } NPF_IfPDH_DS1_DS2_Stats_t;
```

Data structures are:

2.1.3.1.2 DS1-DS2 Current Statistics

```
/*
** Current Basic Statistics
*/

typedef struct {

         NPF_uint32_t  CurrentESs;
                  /* Errored Seconds (ES) */
         NPF_uint32_t  CurrentSESs;
                  /* Severely Errored Seconds (SES) */
         NPF_uint32_t  CurrentSEFSs;
                  /* Severely Errored Framing Seconds (SEFS) */
         NPF_uint32_t  CurrentUASs;
                  /* Unavailable Seconds (UAS) */
         NPF_uint32_t  CurrentCSSs;
                  /* Control Slip Seconds (CSS) */
         NPF_uint32_t  CurrentPCVs;
                  /* Path Coding Violation (PCV) */
         NPF_uint32_t  CurrentLESs;
                  /* Line Errored Seconds (LES) */
         NPF_uint32_t  CurrentBESs;
                  /* Bursty Errored Seconds (BES) */
         NPF_uint32_t  CurrentDMs ;
                  /* Degraded Minutes (DM) */

   } NPF_IfPDH_Stats_BaseCount_t;

/*
** Current Count/Statistics
*/
/*
```

```
** Statistics gathered in the current 15 minutes time interval
*/
typedef struct {

         NPF_IfPDH_Stats_BaseCount_t      Count;
         NPF_uint32_t                     CurrentLCVs;

                                /* Line Coding Violation */
   } NPF_IfPDH_DS1_DS2_CurStats_t;
```

2.1.3.1.3 DS1-DS2 Interval Statistics

```
/*
** Interval Count/Statistics
*/
typedef struct {

/* 15 minute interval number – one of 96 during last 24 hours */

         NPF_uint32_t          IntervalNumber;
         NPF_boolean_t         ValidData;

         NPF_IfPDH_DS1_DS2_CurStats_t     IntervalCurrent;

         } NPF_IfPDH_DS1_DS2_IntervalStats_t;
```

2.1.3.1.4 DS1-DS2 Total Statistics

```
/*
** Total Count/Statistics
*/
typedef struct {

      NPF_IfPDH_DS1_DS2_CurStats_t Total;

      } NPF_IfPDH_DS1_DS2_TotalStats_t;
```

2.1.3.2 DS3 Performance Monitoring Statistics Structures

DS3 Performance Monitoring Statistics are gathered on the following:

- Line Errored Seconds (LES)
- P-bit Errored Seconds (PES)
- P-bit Severely Errored Seconds (PSES)
- C-bit Errored Seconds (CES)
- C-bit Severely Errored Seconds (CSES)
- Unavailable Seconds (UAS)

The structure returned by a Query Statistics function calls would be:

2.1.3.2.1 DS3 Local and Far End Statistics

```
/*
** DS3 Local statistics
*/

typedef struct {

    NPF_IfPDH_DS3_CurStats_t          CurStats;
    NPF_IfPDH_DS3_Intervaltats_t   IntervalStats;
    NPF_IfPDH_DS3_CurStats_t       TotalStats;

    } NPF_IfPDH_DS3_LocalStats_t;

/*
** DS3 Far End statistics
*/

typedef struct {

    NPF_IfPDH_FECurStats_t           CurStats;
    NPF_IfPDH_FEntervalStats_t       IntervalStats;
    NPF_IfPDH_FETotalStats_t         TotalStats;

    } NPF_IfPDH_DS3_FarEndStats_t;
/*
** DS3 Statistics
*/

typedef struct {

    NPF_IfPDH_DS3_LocationStats_t  LocalStats;
    NPF_IfPDH_DS3_LocationStats_t  FarEndStats;

   } NPF_IfPDH_DS3_Stats_t;
```

Data structures are:

2.1.3.2.2 DS3 Current Statistics

```
/*
** Current DS3 Count/Statistics
*/

typedef struct {

         NPF_uint32_t  CurrentPESs;
                  /* P-bit Errored Seconds (PES) */
         NPF_uint32_t  CurrentPSESs;
                  /* P-bit Severely Errored Seconds (PSES) */
         NPF_uint32_t  CurrentSEFSs;
                  /* Severely Errored Framing Seconds (SEFS) */
         NPF_uint32_t  CurrentUASs;
                  /* Unavailable Seconds (UAS) */
         NPF_uint32_t  CurrentLCVs;
                  /* Line Coding Violation */
         NPF_uint32_t  CurrentPCVs;
```

```
                /* P-bit Coding Violation (PCV) */
        NPF_uint32_t  CurrentLESs;
                /* Line Errored Seconds (LES) */
        NPF_uint32_t  CurrentCCVs;
                /* C-bit Coding Violation (CCV) */
        NPF_uint32_t  CurrentCESs;
                /* C-bit Errored Seconds(CES) */
        NPF_uint32_t  CurrentCSESs;
                /* C-bit Severely Errored Seconds(CSES) */

   } NPF_IfPDH_DS3_CurStats_t;
```

2.1.3.2.3 DS3 Interval Statistics

```
/*
** DS3 Interval Count/Statistics
*/
typedef struct {

/*
**  15 minute interval number – most recent of 96 during last 24 hours
*/
      NPF_uint32_t            IntervalNumber;
/*
** Valid or invalid interval
*/
      NPF_boolean_t           ValidData;

      NPF_IfPDH_DS3_CurStats_t IntervalCount;

     } NPF_IfPDH_DS3_IntervalStats_t;
```

2.1.3.2.4 DS3 Total Statistics

```
/*
** DS3 Total Count/Statistics
*/
typedef struct {

    NPF_IfPDH_DS3_CurStats_t TotalCount;

    } NPF_IfPDH_DS3_TotalStats_t;
```

2.1.3.2.5 DS3 Far End Current Statistics

```
/*
** DS3 Far End Current Count/Statistics
*/
typedef struct {
            NPF_uint32_t         FarEndTimeElapsed   ;

            /* time elapsed from measurement at far end */

            NPF_uint32_t         FarEndValidIntervals ;
```

```
            /*
            ** The ID of the 15 minute interval;
            ** Value 0 to 96 if interface.
            */

            NPF_uint32_t         FarEndInvalidIntervals ;

            /*
            ** The number of invalid intervals
            */

          NPF_uint32_t  CurrentCESs;
                   /* C-bit Errored Seconds(CES) */
          NPF_uint32_t  CurrentCSESs;
                   /* C-bit Severely Errored Seconds(CSES) */
          NPF_uint32_t  CurrentCCVs;
                   /* C-bit Coding Violation (CCV) */
          NPF_uint32_t  CurrentUASs;
                   /* Unavailable Seconds (UAS) */

            } NPF_IfPDH_DS3_FECurStats_t;
```

2.1.3.2.6 DS3 Far End Interval Statistics

```
/*
** DS3 Far End Interval Count/Statistics
*/
typedef struct {
/* 15 minute interval number – one of 96 during last 24 hours */

            NPF_uint32_t         FarEndIntervalNumber ;
            NPF_boolean_t        FarEndValidData;
            NPF_IfPDH_DS3_FECurStats_t   FECurStats;

            } NPF_IfPDH_DS3_FEIntervalStats_t;
```

2.1.3.2.7 DS3 Far End Total Statistics

```
/*
** DS3 Far End Total Count/Statistics
*/
typedef struct {

          NPF_IfPDH_DS3_FecCurStats_t  FETotalStats;

            } NPF_IfPDH_DS3_FETotalStats_t;
```

2.2 PDH Completion Callback Type Codes: NPF_IfCallbackType_t

The following Call Back Types are used by PDH interfaces in the **`NPF_IfCallbackType_t`** variable in asynchronous callbacks; this value indicates what function is generating the callback.

```
/*
 *     PDH Completion Callback Types
 */
#define NPF_IF_PDH_LineSpeedAttrSet ((NPF_IF_TYPE_PDH<<16)+1)
#define NPF_IF_PDH_LineTypeAttrSet  ((NPF_IF_TYPE_PDH<<16)+2)
#define NPF_IF_PDH_LineCodingAttrSet ((NPF_IF_TYPE_PDH<<16)+3)
#define NPF_IF_PDH_SendCodeAttrSet ((NPF_IF_TYPE_PDH<<16)+4)
#define NPF_IF_PDH_LoopbackAttrSet ((NPF_IF_TYPE_PDH<<16)+5)
#define NPF_IF_PDH_SignalModeAttrSet ((NPF_IF_TYPE_PDH<<16)+6)
#define NPF_IF_PDH_TxClSrcAttrSet ((NPF_IF_TYPE_PDH<<16)+7)
#define NPF_IF_PDH_FDLinkAttrSet ((NPF_IF_TYPE_PDH<<16)+8)

      /* Fault Management */

#define NPF_IF_PDH_LineStatusQuery ((NPF_IF_TYPE_PDH<<16)+9)
#define NPF_IF_PDH_LoopbackStatusQuery ((NPF_IF_TYPE_PDH<<16)+10)

      /* Performance Monitoring */

#define NPF_IF_PDH_StatisticsQuery ((NPF_IF_TYPE_PDH<<16)+11)
```

The following table summarizes the type of callback, for each of the PDH Interface Management API function call:

Function Name	Type Code
NPF_IfPDH_LineStatusQuery	NPF_IF_PDH_LineStatusQuery
NPF_IfPDH_LoopbackStatusQuery	NPF_IF_PDH_LoopbackStatusQuery
NPF_IfPDH_StatisticsQuery	NPF_IF_PDH_StatisticsQuery

Table 2-1 Table of Function Names and Type Codes

2.2.1 Asynchronous Response Array Element: NPF_IfAsyncResponse_t

The **`NPF_IfAsyncResponse_t`** type is defined in the Core Interface Management IA. This structure contains a union. In this union are pointers to various structures returned by Interface Management API functions. If the PDH interface type is supported, the following must be included in the union within the **`NPF_IfAsyncResponse_t`** structure

```
/*
** The following structures are to be defined in 'npf_if_core.h'
*/


typedef struct NPF_IfPDH_LineStatus NPF_IfPDH_LineStatus_t   ;
typedef struct NPF_IfPDH_LoopbackStatus NPF_IfPDH_LoopbackStatus_t;
typedef struct NPF_IfPDH_Statistics NPF_IfPDH_Statistics_t;



/*
 *     Asynchronous response types for PDH interfaces
 *
```

```
 */

     /* Fault Management Status */

     NPF_IfPDH_LineStatus_t          *if_LineStatus;
     NPF_IfPDH_LoopbackStatus_t      *if_LoopbackStatus;

     /* Performance Monitoring Statistics */

     NPF_IfPDH_Statistics_t          *if_PDHStats;
/*
** End of PDH async response type
*/

/*
** End of definitions to be added also to the 'npf_if_core.h'
*/
```

The following table summarizes the type of callback, and information returned for each of this API function calls:

Type Code	Structure Returned
NPF_IF_PDH_LineStatusQuery	NPF_IfPDH_LineStatus_t
NPF_IF_PDH_LoopbackStatusQuery	NPF_IfPDH_LoopbackStatus_t
NPF_IF_PDH_StatisticsQuery	NPF_IfPDH_Statistics_t

Table 2-2 Table of Callback Codes and Structures Returned by Callbacks

2.3 Interface Management API PDH Error Codes

The following Macro is used to generate values of **`NPF_IfErrorType_t`**.

```
#define NPF_IF_E_PDH_CODE(code) (0x10000+(NPF_IF_TYPE_PDH<<8)+(code))
```

The following PDH Function Calls Error Codes are defined:

```
/*
** PDH Error Codes
*/

/* Invalid PDH Interface Attribute */
#define NPF_IF_E_INVALID_PDH_ATTR NPF_IF_E_PDH_CODE(1)

/* Invalid PDH Interface Binding */
#define NPF_IF_E_INVALID_PDH_BINDING NPF_IF_E_PDH_CODE(2)

/* Invalid PDH Line Speed */
#define NPF_IF_E_INVALID_PDH_LINESPEED NPF_IF_E_PDH_CODE(3)
```

```
/* Invalid PDH Line Type */
#define NPF_IF_E_INVALID_PDH_LINETYPE NPF_IF_E_PDH_CODE(4)
/* Invalid PDH Line Coding */
#define NPF_IF_E_INVALID_PDH_LINECODING NPF_IF_E_PDH_CODE(5)

/* Invalid PDH Send Code */
#define NPF_IF_E_INVALID_PDH_SENDCODE NPF_IF_E_PDH_CODE(6)

/* Invalid PDH Loopback Configuration */
#define NPF_IF_E_INVALID_PDH_LOOPBACK NPF_IF_E_PDH_CODE(7)

/* Invalid PDH Signal Mode */
#define NPF_IF_E_INVALID_PDH_SIGNALMODE NPF_IF_E_PDH_CODE(8)

/* Invalid PDH Transmit Clock Source */
#define NPF_IF_E_INVALID_PDH NPF_IF_E_PDH_CODE(10)

/* Invalid PDH Facilities Data Link */
#define NPF_IF_E_INVALID_PDH_FDLINK NPF_IF_E_PDH_CODE(10)

/*
** End of PDH Error Codes
*/
```

2.4 Interface Managament API PDH Events

The PDH events are of two categories: Fault Management events and Performance Monitoring events. The PDH events must be added to the list of events define in the "core" IM API.

2.4.1 PDH Fault Management Events

The Fault Management events for PDH interfaces are:

- Bipolar Violation (BPV) Error Event
- Excessive Zeroes (EXZ) Error Event
- Line Coding Violation (LCV) Error Event
- Path Coding Violation (PCV) Error Event
- Controlled Slip (CS) Error Event

```
/*
** PDH Interface Management Fault Management Events
*/

/*
 */
```

```
#define NPF_IF_PDH_BPV NPF_TYPE_PDH<<16) + 1)
                                    /* Bipolar Violation Error Event */
#define NPF_IF_PDH_EXZ NPF_TYPE_PDH<<16) + 2)
                                    /* Excessive Zeroes Error Event */
#define NPF_IF_PDH_LCV NPF_TYPE_PDH<<16) + 3)
                                    /* Line Coding Violation Event */
#define NPF_IF_PDH_PCV NPF_TYPE_PDH<<16) + 4)
                                    /* Path Coding Violation Event */
#define NPF_IF_PDH_CS NPF_TYPE_PDH<<16) + 5)
                                    /* Controlled Slip Error Event */
#define NPF_IF_PDH_OOF NPF_TYPE_PDH<<16) + 6)
                                    /* Out Of Frame */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 7)
                                    /* Alarm Indication Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 8)
                                    /* RDI Failure (new name for RAI) */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 9)
                                    /* NCI Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 10)
                                    /* TIM Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 11)
                                    /* DEG Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 12)
                                    /* UNEQ Failure */




#define NPF_IF_PDH_LOF NPF_TYPE_PDH<<16) + 13)
                                    /* PDH Interface Loss Of Frame */
#define NPF_IF_PDH_LOS NPF_TYPE_PDH<<16) + 14)
                                    /* PDH Interface Loss Of Signal */
#define NPF_IF_PDH_LPF NPF_TYPE_PDH<<16) + 15)
                                    /* PDH Interface Loopback Pseudo-Failure */
#define NPF_IF_PDH_LOMF NPF_TYPE_PDH<<16) + 16)
                                    /* PDH Interface Loss of Multi-Frame Failure */
#define NPF_IF_PDH_RAI NPF_TYPE_PDH<<16) + 17)
                                    /* PDH Interface Remote Alarm Indication */
```

2.4.2 PDH Performance Monitoring Events

The PDH Performance Monitoring events are triggered by PDH Performance Monitoring alert notifications sent by the PDH layer upon detecting a crossing of the threshold level

```
/*
**  PDH Performance Monitoring Events
*/

typedef enum {

#define NPF_IF_PDH_ES NPF_TYPE_PDH<<16) + 100 + 1)
                                        /* Errored Seconds */
#define NPF_IF_PDH_SES NPF_TYPE_PDH<<16) + 100 + 2)
                                        /* Severe Errored Seconds */
#define NPF_IF_PDH_SEFS NPF_TYPE_PDH<<16) + 100 + 3)
                                        /* Severely Errored Framing Seconds */
#define NPF_IF_PDH_CV NPF_TYPE_PDH<<16) + 100 + 4)
                                        /* Coding Violation */
#define NPF_IF_PDH_US NPF_TYPE_PDH<<16) + 100 + 5)
                                        /* Unavailable Seconds */
```

2.4.3 PDH Event Notification

PDH Event notification will be performed through the existing IM Core API NPF Event data structures. The following must be added to the union within the NPF_IfEvent_Data_t structure:

```
NPF_IfPDH_LineStatus  *PDHLineStatus; /* Line Status returned */
```

3 Function Definitions

The following existing functions can be used with PDH interfaces:

- **NPF_IfCreate()**
- **NPF_IfDelete()**
- **NPF_IfCreateAndSet()**
- **NPF_IfGenericStatsGet()**
- **NPF_IfAttrSet()**
- **NPF_IfEnable()**
- **NPF_IfDisable()**
- **NPF_IfOperStatusGet()**

3.1 Generic Function Calls

NPF_IfAttrSet

Set all Interface Attributes

Syntax

```
NPF_error_t  NPF_IfAttrSet(
      NPF_IN NPF_callbackHandle_t    if_cbHandle,
      NPF_IN NPF_correlator_t        if_cbCorrelator,
      NPF_IN NPF_errorReporting_t    if_errorReporting,
      NPF_IN NPF_uint32_t            n_handles,
      NPF_IN NPF_IfHandle_t         *if_HandleArray,
      NPF_IN NPF_IfGeneric_t       *if_StructArray);
```

Description of function

This function sets all the attributes of one or more interfaces, from the contents of an array of structures passed by the caller, as defined in **NPF_IfGeneric_t**. Ownership of the structure memory remains with the caller (the API implementation must copy all needed contents before returning). Any single attribute can be set with its own function call; this function is included as a way to set multiple attributes atomically and efficiently. Note: the number of **NPF_IfGeneric_t** structures and the number of interface handles in the two arrays must be the same, equal to the **n_handles** argument. This function sets a *different* set of attributes for each named interface. The Interface Handle value identifies the interface to be modified; the Interface ID value in the **NPF_IfGeneric_t** structure is ignored.

Input Parameters

- **if_cbHandle**: the registered callback handle.
- **if_cbCorrelator**: the application's context for this call.

- **`if_errorReporting`**: the desired callback.
- **`n_handles`**: the number of interfaces to set attributes for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- **`if_StructArray`**: pointer to a structurean array of structures containing the new interface attributes.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_ATTRIBUTE`**: An attribute (other than those mentioned below) was invalid.

Generic Interface Errors:

- **`NPF_IF_E_INVALID_HANDLE`**: if_Handle is null or invalid.
- **`NPF_IF_E_INVALID_SPEED`**: Invalid interface speed parameter.
- **`NPF_IF_E_INVALID_IF_TYPE`**: Invalid interface type code.
- **`NPF_IF_E_INVALID_ADMIN_STATUS`**: Invalid administrative status code.

PDH Interface Errors:

- **`NPF_IF_E_INVALID_PDH_ATTR :`** Invalid PDH Interface Attribute.

Asynchronous response

A total of **`n_handles`** asynchronous responses (**`NPF_IfAsyncResponse_t`**) will be passed to the callback function, in one or more invocations. Each response contains an interface handle and a success code or a possible error code for that interface. The union in the callback response structure is unused.

NPF_IfCreateAndSet

Create an Interface and Set All of its Attributes

Syntax

```
NPF_error_t  NPF_IfCreateAndSet(
      NPF_IN NPF_callbackHandle_t if_cbHandle,
      NPF_IN NPF_correlator_t if_cbCorrelator,
      NPF_IN NPF_errorReporting_t if_errorReporting,
      NPF_IN NPF_uint32_t n_if,
      NPF_IN NPF_IfGeneric_t *if_StructArray);
```

Description of function

This function simultaneously creates and sets all the attributes of one or more interfaces, from the contents of an array of structures passed by the caller (**`NPF_IfGeneric_t`**). Each interface is created with a *different* set of attributes. Ownership of the structure memory remains with the caller (the API implementation must copy all contents before returning). Each instance of the

NPF_IfGeneric_t structure must contain a different, nonzero Interface ID value, and none may be the same as that of an existing interface.

Input Parameters

- **if_cbHandle**: the registered callback handle.
- **if_cbCorrelator**: the application's context for this call.
- **if_errorReporting**: the desired callback.
- **n_if**: the number of interfaces to set attributes for.
- **if_StructArray**: pointer to an array of structures containing the new interface attributes.

Output Parameters

None

Asynchronous Error Codes

- **NPF_NO_ERROR**: Operation successful.
- **NPF_E_RESOURCE_EXISTS**: an interface with the same Interface ID value already exists; its handle is returned in the callback, and no new interface is created.
- **NPF_IF_E_INVALID_ATTRIBUTE**: An attribute (other than those mentioned below) was invalid.

Generic Interface Errors:

- **NPF_IF_E_INVALID_HANDLE** : if_Handle is null or invalid.
- **NPF_IF_E_INVALID_SPEED**: Invalid interface speed parameter.
- **NPF_IF_E_INVALID_IF_TYPE**: Invalid interface type code.
- **NPF_IF_E_INVALID_ADMIN_STATUS**: Invalid administrative status code.

PDH Interface Errors:

- **NPF_IF_E_INVALID_PDH_ATTR :** Invalid PDH Interface Attribute..

Asynchronous response

A total of **n_if** asynchronous responses (**NPF_IfAsyncResponse_t**) will be passed to the callback function, in one or more invocations. Each response contains the new interface handle and a success code or a possible error code if an interface could not be created or any attributes could not be set. Responses are linked to interface attributes in the following way: for each response, the union in the response structure contains the corresponding index of the **if_StructArray** element that contained its attributes. For example, the response for the first array element will include an Interface Handle and an **arrayIndex** value of zero; the response for the tenth array element an **arrayIndex** of 9, and so on.

NPF_IfBind

Bind Interfaces

Syntax

```
NPF_error_t NPF_IfBind(
```

```
NPF_IN NPF_callbackHandle_t    if_cbHandle,
NPF_IN NPF_correlator_t        if_cbCorrelator,
NPF_IN NPF_errorReporting_t    if_errorReporting,
NPF_IN NPF_uint32_t            nbinds,
NPF_IN NPF_IfBinding_t         *if_bindArray);
```

Description

This function binds one or more pairs of PDH interfaces in parent-child relationships. Each binding associates two interfaces with each other, one as parent, and one as child. Multiple bindings can be made in a single call. An interface can have multiple parents; it can also have multiple children. An interface can be at the same time the parent of one and the child of another. An implementation SHOULD return an error if cycles occur (e.g. an interface is the child of one of its own children: "I'm my own grandpa"). An implementation MAY limit how many associations an interface can have, or restrict the depth of the hierarchy.

Bindings have the following characteristics:

- Adding a PDH parent interface to another PDH interface means that the parent interface is a lower layer in the PDH hierarchy, while the child is a higher layer in the PDH hierarchy.
- Not all combinations of PDH parent-child interfaces are valid. An implementation MUST return a binding error if the PDH binding combination is invalid.
- The following PDH interface bindings are valid (see **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, **Error! Reference source not found.**, and **Error! Reference source not found.**):
    - PDH DS3, T3, or E3 interface parent to DS2, T2, or E2 PDH interfaces, or DS1, T1, or E1 interfaces.
    - PDH DS2, T2, or E2 interface parent to DS1, T1, or E1 interfaces.
    - PDH DS1, T1, or E1 interface parent to DS0 interfaces.

- Removing a binding does not result in either interface being deleted.

Input Parameters

- **`if_cbHandle:`** the registered callback handle.
- **`if_cbCorrelator`**: the application's context for this call.
- **`if_errorReporting`**: the desired callback.
- **`nbinds`**: number of bindings in the array.
- **`if_bindArray`**: pointer to an array of interface handle parent/child bindings.

Output Parameters

None

Asynchronous Error Codes

- **NPF_NO_ERROR**: Operation successful.
- **NPF_IF_E_INVALID_CHILD_HANDLE**: Child Handle is null or invalid; no binding done.
- **NPF_IF_E_INVALID_PARENT_HANDLE**: Parent Handle is null or invalid; no binding done.
- **NPF_IF_E_INVALID_PARAM**: Binding failed. No binding
- **NPF_IF_E_CIRCULAR_BINDING**: An interface would exist more than once in its own parent/child hierarchy. Binding failed; no binding done.

PDH Interface Errors:

- **NPF_IF_E_INVALID_PDH_BINDING:** Invalid PDH Interface Binding . Binding failed; no binding done.

Asynchronous Response

A total of **n_binds** asynchronous responses (**NPF_IfAsyncResponse_t**) will be passed to the callback function, in one or more invocations. Each response contains the parent interface handle and a possible error code. The particular binding to which the response code pertains is identified in the callback by the two handles: the parent handle is in the usual **ifHandle** position, and the child handle is in the union part of the callback structure.

In addition, the following new functions are defined:

3.2 Attribute Setting Function Calls

All PDH Interface Attributes can be set of Generic Interface Managament API calls.

3.3 Fault Management Function Calls

3.3.1 NPF_IfPDH_LineStatusQuery

Function to Get Line Status for a PDH Interface

Syntax

```
NPF_error_t  NPF_IfPDH_LineStatusQuery(
      NPF_IN NPF_callbackHandle_t    if_cbHandle,
```

```
        NPF_IN NPF_correlator_t       if_cbCorrelator,
        NPF_IN NPF_errorReporting_t   if_errorReporting,
        NPF_IN NPF_IfHandle_t         if_Handle);
```

Description of function

This function returns, via a callback, a pointer to a PDH line status structure.

Input Parameters

- **`if_cbHandle`**: the registered callback handle.
- **`if_cbCorrelator`**: the application's context for this call.
- **`if_errorReporting`**: the desired callback.
- **`if_Handle`**: the handle of an interface of type PDH (one of the PDH interface types)

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: **`if_Handle`** is null or invalid, or is not a PDH interface.

Asynchronous response

An asynchronous responses (**`NPF_IfAsyncResponse_t`**) will be passed to the callback function, in one invocation. The response contains the interface handle of the PDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a pointer to a PDH Line Status structure.

3.3.2 NPF_IfPDH_LoopbackStatusQuery

Function to Get Loopback Status for a PDH Interface

Syntax

```
NPF_error_t  NPF_IfPDH_LineStatusQuery(
      NPF_IN NPF_callbackHandle_t   if_cbHandle,
      NPF_IN NPF_correlator_t       if_cbCorrelator,
      NPF_IN NPF_errorReporting_t   if_errorReporting,
      NPF_IN NPF_IfHandle_t         if_Handle);
```

Description of function

This function returns, via a callback, a pointer to a PDH loopback status structure.

Input Parameters

- **`if_cbHandle`**: the registered callback handle.
- **`if_cbCorrelator`**: the application's context for this call.
- **`if_errorReporting`**: the desired callback.
- **`if_Handle`**: the handle of an interface of type PDH (one of the PDH interface types)

Output Parameter

None

Asynchronous Error Codes

- **NPF_NO_ERROR**: Operation successful.
- **NPF_IF_E_INVALID_HANDLE**: **if_Handle** is null or invalid, or is not a PDH interface.

Asynchronous response

An asynchronous responses (**NPF_IfAsyncResponse_t**) will be passed to the callback function, in one invocation. The response contains the interface handle of the PDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a pointer to a PDH Loopback Status structure.

3.4 Performance Monitoring Function Calls

3.4.1 NPF_IfPDH_StatisticsQuery

Function to Get Statistics of a PDH Interface

Syntax

```
NPF_error_t  NPF_IfPDH_StatisticsQuery(
      NPF_IN NPF_callbackHandle_t    if_cbHandle,
      NPF_IN NPF_correlator_t        if_cbCorrelator,
      NPF_IN NPF_errorReporting_t    if_errorReporting,
      NPF_IN NPF_uint32_t            n_handles,
      NPF_IN NPF_IfHandle_t         *if_HandleArray);
```

Description of function

This function returns, via a callback, a pointer to a PDH Interface Performance Monitoring statistics structure containing the statistics values for one or more indicated interfaces.

Input Parameters

- **if_cbHandle**: the registered callback handle.
- **if_cbCorrelator**: the application's context for this call.
- **if_errorReporting**: the desired callback.
- **n_handles**: the number of interfaces to get statistics for.
- **if_HandleArray**: pointer to an array of interface handles.

Output Parameters

None

Asynchronous Error Codes

- **NPF_NO_ERROR**: Operation successful.
- **NPF_IF_E_INVALID_HANDLE:** An if_Handle is null or invalid.

Asynchronous response

A total of **n_handles** asynchronous responses (**NPF_IfAsyncResponse_t**) will be passed to the callback function, in one or more invocations. Each response contains an interface handle or a possible error code. If the error code indicates success, the union in the callback response structure contains a pointer to the **NPF_IfPDH_Stats_t** structure for that interface.

4 Summary

4.1 Summary of API Functions

The following is a summary table of the PDH Function Calls:

Function Name	Required?
NPF_IfPDH_LineStatusQuery()	Yes
NPF_IfPDH_LoopbackStatusQuery()	Yes
NPF_IfPDH_StatisticsQuery()	Yes

Table 4-1 Summary of Function Calls

4.2 Summary of API Functions and Input Data Structures

The following table summarizes the Function Calls and Data Structures that each function accepts as input parameter:

Function Name	Input Data Structure

Table 4-2 Summary of Function Calls and Input Data Structures

5 References

- NPF Software API Conventions IA
- [NPF2002.471.04] NPF Interface Management API IA revision 2.0
- IETF – RFC2494 :Definitions of Managed Objects for the DS0 Interface Type
- IETF – RFC2495 :Definitions of Managed Objects for the DS1-DS2 Interface Type
- IETF – RFC2496 :Definitions of Managed Objects for the DS3 Interface Type
- ANSI T1.102 «Telecommunications – Elecrical Digital Hierarchy – Electrical interfaces »
- ANSI T1.107 « Digital Hierarchy –Formats Specifications»
- ANSI T1.404 « Network-to-Customer Installation – DS3 Metallic Interface Specification »
- ITU-T G.751
- TelCordia – GR-253
- ITU-T G.703
  "Physical/electrical characteristics of hierarchical digital interfaces"
- ITU-T G.704
  "Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels"

- ITU-T G.705
  "Characteristics of plesiochronous digital hierarchy (PDH) equipment functional blocks"
- ITU-T G.742
  "Second order digital multiplex equipment operating at 8448 kbit/s and using positive justification"
- ITU-T G.751
  "Digital multiplex equipments operating at the third order bit rate of 34 368 kbit/s and the fourth order bit rate of 139 264 kbit/s and using positive justification"
- ITU-T G.804
  "ATM cell mapping into plesiochronous digital hierarchy (PDH)"
- ITU-T G.806
  "Characteristics of transport equipment - Description methodology and generic functionality"
- ITU-T G.832
  "Transport of SDH elements on PDH networks - Frame and multiplexing structures"
- ATIS T1.231
  "Digital Hierarchy - Layer 1 In-Service Digital Transmission Performance Monitoring"
- ETSI EN 300 417 series:
  "Transmission and Multiplexing (TM); Generic requirements of transport functionality of equipment"
- ETSI EN 300 417-1-1
  "Part 1-1: Generic processes and performance"
- ETSI EN 300 417-2-1
  "Part 2-1: Synchronous Digital Hierarchy (SDH) and Plesiochronous Digital Hierarchy (PDH) physical section layer functions"
- ETSI EN 300 417-5-1
  "Part 5-1: Plesiochronous Digital Hierarchy (PDH) path layer functions"

6 Revision History

V00	04/21/2004	Merge contribution npf2004.080 with contribution npf2004.122 Generate new data structures and function calls.
V01	04/22/2004	Add .h file in Appendix. Added subsection in Introduction.
V02	07/05/2004	Change to new IM API format
V03	08/30/2004	Corrections to text, Appendix B, and Appendix A (make sure code compiles)
V04	08/31/04	Change “BINDING” error to “PDH_BINDING”
V05	09/02/04	Additional corrections to Appendix A
V06	10/08/04	Straw Ballot Comment Resolution text

APPENDIX A NPF_IF_PDF.H

```
/* NPF_IF_PDH.h */

/*
**  This header file defines typedefs, constants, and functions
**  that apply to the NPF PDH Interface Management API.
**  It is defined based on the Interface Management API structures.
**  It contains some of the structures from "npf_if.h" file, which were
**  extended with SONET/SDH API Data Structures definitions.
**
**
*/


#ifndef __NPF_IF_PDH_H_
#define __NPF_IF_PDH_H_

#ifdef __cplusplus
extern "C"  {
#endif

/*
**  *** PDH Definitions
*/

/*
** +++ Interface Type definitions
*/

#define     NPF_IF_TYPE_PDH   11          /* PDH interface */

  typedef enum{

    NPF_IF_PDH_TYPE_DS0        = 1,       /* US Standard */
    NPF_IF_PDH_TYPE_DS0_BUNDLE = 2,       /* US Standard */
    NPF_IF_PDH_TYPE_DS1        = 3,       /* US standard */
    NPF_IF_PDH_TYPE_DS2        = 4,       /* US standard */
    NPF_IF_PDH_TYPE_DS3        = 5,       /* US standard */

    NPF_IF_PDH_TYPE_T1         = 6,       /* US standard */
    NPF_IF_PDH_TYPE_T2         = 7,       /* US standard */
    NPF_IF_PDH_TYPE_T3         = 8,       /* US standard */

    NPF_IF_PDH_TYPE_E1         = 9,       /* European standard */
    NPF_IF_PDH_TYPE_E2         = 10,      /* European standard */
    NPF_IF_PDH_TYPE_E3         = 11,      /* European standard */
    NPF_IF_PDH_TYPE_E3         = 12,      /* European standard */

    NPF_IF_PDH_TYPE_J1         = 13,      /* Japanese standard 1 */
    NPF_IF_PDH_TYPE_J2         = 14       /* Japanese standard 2 */

      }NPF_IfPDH_Type_t;
```

```
/*
** Line Speed
*/


  typedef enum {

NPF_IfPDH_DS1 = 1,   /* DS1 - 1.544 Mbit/sec */
NPF_IfPDH_E1 = 2,    /* E1  - 2.048 Mbit/sec */
NPF_IfPDH_DS2 = 3,   /* DS2 - 6.312 Mbit/sec */
NPF_IfPDH_E2 = 4,    /* E2  - 8.448 Mbit/sec */
NPF_IfPDH_DS3 = 5,   /* DS3 - 44.736 Mbit/sec */
NPF_IfPDH_E3 = 6,    /* E3 - 34.368 Mbit/sec */
NPF_IfPDH_E3 = 7     /* E4 - 139.264 Mbit/sec */
} NPF_IfPDH_LineSpeed_t;


/*
** Line Type
*/
typedef enum {

NPF_IF_PDH_LINE_TYPE_OtherType  = 0,
            /* Type other */
NPF_IF_PDH_LINE_TYPE_DS1_ESF = 1,
            /*  Extended SuperFrame DS1 (T1.107) */
NPF_IF_PDH_LINE_TYPE_DS1_D4 = 2,
            /*  AT&T D4 format DS1 (T1.107) */
NPF_IF_PDH_LINE_TYPE_E1 = 3,
            /* ITU-T Recommendation G.704 */
NPF_IF_PDH_LINE_TYPE_E1_CRC = 4,
            /* ITU-T Recommendation G.704 */
NPF_IF_PDH_LINE_TYPE_E1_MF = 5,
            /* G.704 with TS16 multi-framing enabled */
NPF_IF_PDH_LINE_TYPE_E1_CRCMF = 6,
            /* G.704 with TS16 multi-framing enabled */
NPF_IF_PDH_LINE_TYPE_DS1_Unframed = 7,
            /* DS1 with no framing */
NPF_IF_PDH_LINE_TYPE_E1_Unframed = 8,
            /* E1 with no framing */
      NPF_IF_PDH_LINE_TYPE_DS2_M12 = 9,
                  /* DS2 frame format (T.107) */
NPF_IF_PDH_LINE_TYPE_E2 = 10,
            /* E2 Frame format (G.704) */
NPF_IF_PDH_LINE_TYPE_DS3_M23 = 11,
            /* ANSI T1.107-1988 */
NPF_IF_PDH_LINE_TYPE_DS3_SYNTRAN = 12,
            /* ANSI T1.107-1988 */
NPF_IF_PDH_LINE_TYPE_DS3_CbitParity = 13,
            /* ANSI T1.107a-1990 */
NPF_IF_PDH_LINE_TYPE_DS3_ClearChannel = 14,
/* ANSI T1.102-1987 */
NPF_IF_PDH_LINE_TYPE_E3_Framed = 15,
            /* CCITT G.751*/
NPF_IF_PDH_LINE_TYPE_E3_Plcp = 16,
            /* ETSI T/NA (1998) */
NPF_IF_PDH_LINE_TYPE_E3_G832 = 17,
```

```
            /* */
NPF_IF_PDH_LINE_TYPE_E4 = 18,
            /* */
NPF_IF_PDH_LINE_TYPE_E4_G832 = 19
            /* */

            /* Type other */

} NPF_IfPDH_LineType_t;

/*
** Line Coding
*/
typedef enum {
NPF_IF_PDH_LineCoding_Other = 0,

NPF_IF_PDH_LineCoding_DS1_JBZS  = 1,
NPF_IF_PDH_LineCoding_DS1_B8ZS  = 2,
NPF_IF_PDH_LineCoding_DS1_HDB3  = 3,
NPF_IF_PDH_LineCoding_DS1_ZBTSI  = 4,
NPF_IF_PDH_LineCoding_DS1_AMI = 5,
NPF_IF_PDH_LineCoding_DS1_B6ZS = 6,
/* DS3 Line Coding */
NPF_IF_PDH_LineCoding_DS3_B3ZS = 8,
NPF_IF_PDH_LineCoding_E3_HDB3 = 9

}NPF_IfPDH_LineCoding_t;


/*
** Send Code
*/

typedef enum {

NPF_IF_PDH_SendNoCode = 1,

NPF_IF_PDH_SendLineCode = 2,
NPF_IF_PDH_SendPayloadCode = 3,
NPF_IF_PDH_SendResetCode = 4,

/* */

NPF_IF_PDH_DS1_SendQRS = 5,
NPF_IF_PDH_DS1_Send511Pattern = 6,
NPF_IF_PDH_DS1_Send3in24Pattern = 7,
NPF_IF_PDH_DS1_SendOtherTestPattern = 8,

/* */

NPF_IF_PDH_DS3_SendDS1LoopCode = 13,
NPF_IF_PDH_DS3_SendTestPattern = 14

} NPF_IfPDH_SendCode_t;


/*
```

```
** Loopback
*/

typedef enum {

      NPF_IF_LineLoop  = 1,
    NPF_IF_PayloadLoop = 2,
      NPF_IF_Inward = 3,
      NPF_IF_Dual = 4,
      NPF_IF_NoLoopback = 5,
      NPF_IF_OtherLoop = 6

} NPF_IfPDH_Loopback_t;



/*
** SignalMode
*/
typedef enum {

NPF_IF_PDH_None = 1,
NPF_IF_PDH_RobbedBit = 2,
NPF_IF_PDH_BitOriented = 3,
NPF_IF_PDH_MessageOriented = 4,
NPF_IF_PDH_SignalMode_Other = 5

} NPF_IfPDH_SignalMode_t;


/*
** Transmit Clock Source
*/
typedef enum{

NPF_IF_PDH_loopTiming = 1,
NPF_IF_PDH_localTiming = 2,
NPF_IF_PDH_throughtiming = 3

} NPF_IfPDH_TxClSrc_t;

/*
**  Facilities Data Link
*/
typedef enum{

NPF_IF_PDH_Other = 1,
NPF_IF_PDH_Dsx1ANSIT1403 = 2,
NPF_IF_PDH_ATT54016 = 3,
NPF_IF_PDH_FDLNone = 4

} NPF_IfPDH_FDLink_t;

/*
** Line Status
*/
typedef enum {
```

```
NPF_IF_PDH_NoAlarm  = 1,
/* No alarm present */
NPF_IF_PDH_DS1_RcvFarEndLOF = 2,
/* Far end LOF (a.k.a., Yellow Alarm) */
NPF_IF_PDH_DS1_XmtFarEndLOF = 3,
/* Near end sending LOF Indication */
NPF_IF_PDH_RcvAIS = 4,
/* Far end sending AIS */
NPF_IF_PDH_XmtAIS = 5,
  /* Near end sending AIS */
NPF_IF_PDH_LOF = 6,
   /* Near end LOF (a.k.a., Red Alarm) */
NPF_IF_PDH_LOS = 7,
      /* Near end Loss Of Signal */
NPF_IF_PDH_LoopbackState = 8,
/* Near end is looped */
NPF_IF_PDH_DS1_T16AIS = 9,
    /* E1 TS16 AIS */
NPF_IF_PDH_DS1_RcvFarEndLOMF = 10,
/* Far End Sending TS16 LOMF */
NPF_IF_PDH_DS1_XmtFarEndLOMF = 11,
/* Near End Sending TS16 LOMF */
NPF_IF_PDH_RcvTestCode   = 12,
/*  Near End detects a test code */
NPF_IF_PDH_OtherFailure  = 13,
/* any line status not defined here */
NPF_IF_PDH_UnavailSigState = 14,
/* Near End in Unavailable Signal State */
NPF_IF_PDH_NetEquipOOS = 15,
/* Carrier Equipment Out of Service */
NPF_IF_PDH_DS2_RcvPayloadAIS = 16,
/* DS2 Payload AIS */
NPF_IF_PDH_DS2_PerfThreshold = 17,
/* DS2 Performance Threshold Exceeded */
NPF_IF_PDH_DS3_RcvRAIFailure  = 18,
/*  Receiving Yellow/RemoteAlarm Indication */
NPF_IF_PDH_DS3_XmitRAIAlarm  = 19
/* Transmit Yellow/RemoteAlarm Indication */
      } NPF_IfPDH_LineStatusBits_t;


struct NPF_IfPDH_LineStatus{
       NPF_uint16_t    n_Statusbits;
       NPF_IfPDH_LineStatusBits_t  *Statusvalue;
};
/*
** Loopback Status
*/
typedef enum {

NPF_IF_PDH_NoLoopback = 1,
NPF_IF_PDH_NearEndPayloadLoopback = 2,
NPF_IF_PDH_NearEndLineLoopback = 3,
NPF_IF_PDH_NearEndOtherLoopback = 4,
NPF_IF_PDH_NearEndInwardLoopback = 5,
NPF_IF_PDH_FarEndPayloadLoopback = 6,
NPF_IF_PDH_FarEndLineLoopback = 7
```

```
} NPF_IfPDH_LoopbackStatusBits_t;

struct NPF_IfPDH_LoopbackStatus{
       NPF_uint16_t    n_bits;
       NPF_IfPDH_LoopbackStatusBits_t  *value;
};


/*
**  PDH Interface Attributes
*/
struct NPF_IfPDH{

 NPF_uint32_t     port;       /* Port number */

 NPF_IfPDH_Type_t  PDHType;    /* PDH Interface type */

 NPF_IfPDH_LineSpeed_t        LineSpeed;
 NPF_IfPDH_LineType_t   LineType;
 NPF_IfPDH_LineCoding_t LineCode;
 NPF_IfPDH_SendCode_t   SendCode;
 NPF_IfPDH_Loopback_t         Loopback;
/**/
 NPF_IfPDH_SignalMode_t SignalMode;
 NPF_IfPDH_TxClSrc_t    XMTClockSource;
 NPF_IfPDH_FDLink_t     FacilDataLink;
/**/
  NPF_uint32_t    InvalidIntervals;
  NPF_uint32_t    LineLength;
  NPF_uint32_t    LineStatusLastChange;
/**/
 NPF_uint32_t     TimeElapsed;
 NPF_uint32_t     ValidIntervals;
/**/
 NPF_IfPDH_LineStatus_t       LineStatus;
 NPF_IfPDH_LoopbackStatus_t   LoopbackStatus;
/**/
    };


/*
** +++ PDH Statistics
*/

/*
** --- DS1_DS2 Current Basic Statistics
*/

typedef struct {

NPF_uint32_t  CurrentESs;
/* Errored Seconds (ES) */
NPF_uint32_t  CurrentSESs;
/* Severely Errored Seconds (SES) */
NPF_uint32_t  CurrentSEFSs;
/* Severely Errored Framing Seconds (SEFS) */
```

```
NPF_uint32_t  CurrentUASs;
/* Unavailable Seconds (UAS) */
NPF_uint32_t  CurrentCSSs;
/* Control Slip Seconds (CSS) */
NPF_uint32_t  CurrentPCVs;
/* Path Coding Violation (PCV) */
NPF_uint32_t  CurrentLESs;
/* Line Errored Seconds (LES) */
NPF_uint32_t  CurrentBESs;
/* Bursty Errored Seconds (BES) */
NPF_uint32_t  CurrentDMs ;
/* Degraded Minutes (DM) */

} NPF_IfPDH_Stats_BaseCount_t;

/*
** Current Count/Statistics
*/

/*
** Statistics gathered in the current 15 minutes time interval
*/
typedef struct {

NPF_IfPDH_Stats_BaseCount_t    Count;
NPF_uint32_t                   CurrentLCVs;

                        /* Line Coding Violation */
} NPF_IfPDH_DS1_DS2_CurStats_t;




/*
** Interval Count/Statistics
*/
typedef struct {

/* 15 minute interval number - one of 96 during last 24 hours */

NPF_uint32_t            IntervalNumber;
NPF_boolean_t           ValidData;

NPF_IfPDH_DS1_DS2_CurStats_t   IntervalCurrent;

} NPF_IfPDH_DS1_DS2_IntervalStats_t;


/*
** DS1-DS2 Local statistics
*/

typedef struct {

    NPF_IfPDH_DS1_DS2_CurStats_t     CurStats;
    NPF_IfPDH_DS1_DS2_IntervalStats_t  IntervalStats;
```

```
    NPF_IfPDH_DS1_DS2_CurStats_t       TotalStats;

 } NPF_IfPDH_DS1_DS2_LocalStats_t;



/*
** DS1-DS2 Statistics
*/

typedef struct {

    NPF_IfPDH_DS1_DS2_LocalStats_t  LocalStats;
    NPF_IfPDH_DS1_DS2_LocalStats_t  FarEndStats;

} NPF_IfPDH_DS1_DS2_Stats_t;


/*
** Total DS1_DS2 Count/Statistics
*/
typedef struct {

      NPF_IfPDH_DS1_DS2_CurStats_t Total;

} NPF_IfPDH_DS1_DS2_TotalStats_t;


/*
** ---- DS3 Local statistics
*/

/*
** Current DS3 Count/Statistics
*/

typedef struct {

NPF_uint32_t  CurrentPESs;
/* P-bit Errored Seconds (PES) */
NPF_uint32_t  CurrentPSESs;
/* P-bit Severely Errored Seconds (PSES) */
NPF_uint32_t  CurrentSEFSs;
/* Severely Errored Framing Seconds (SEFS) */
NPF_uint32_t  CurrentUASs;
/* Unavailable Seconds (UAS) */
NPF_uint32_t  CurrentLCVs;
            /* Line Coding Violation */
NPF_uint32_t  CurrentPCVs;
/* P-bit Coding Violation (PCV) */
NPF_uint32_t  CurrentLESs;
/* Line Errored Seconds (LES) */
NPF_uint32_t  CurrentCCVs;
/* C-bit Coding Violation (CCV) */
NPF_uint32_t  CurrentCESs;
/* C-bit Errored Seconds(CES) */
NPF_uint32_t  CurrentCSESs;
```

```
/* C-bit Severely Errored Seconds(CSES) */

} NPF_IfPDH_DS3_CurStats_t;


/*
** DS3 Interval Count/Statistics
*/
typedef struct {

/*
**  15 minute interval number - most recent of 96 during last 24 hours
*/
      NPF_uint32_t            IntervalNumber;
/*
** Valid or invalid interval
*/
      NPF_boolean_t           ValidData;

      NPF_IfPDH_DS3_CurStats_t IntervalCount;

     } NPF_IfPDH_DS3_IntervalStats_t;

/*
** DS3 Local statistics
*/


typedef struct {

    NPF_IfPDH_DS3_CurStats_t        CurStats;
    NPF_IfPDH_DS3_IntervalStats_t   IntervalStats;
    NPF_IfPDH_DS3_CurStats_t        TotalStats;

 } NPF_IfPDH_DS3_LocalStats_t;


/*
** DS1-DS2 Statistics
*/

typedef struct {

    NPF_IfPDH_DS3_LocalStats_t  LocalStats;

    NPF_IfPDH_DS3_LocalStats_t  FarEndStats;


} NPF_IfPDH_DS3_Stats_t;



/*
** DS3 Total Count/Statistics
*/
typedef struct {
```

```
    NPF_IfPDH_DS3_CurStats_t TotalCount;

    } NPF_IfPDH_DS3_TotalStats_t;

/*
** DS3 Far End Current Count/Statistics
*/
typedef struct {
NPF_uint32_t         FarEndTimeElapsed   ;

/* time elapsed from measurement at far end */

NPF_uint32_t         FarEndValidIntervals ;

/*
** The ID of the 15 minute interval;
** Value 0 to 96 if interface.
*/

NPF_uint32_t         FarEndInvalidIntervals ;

/*
** The number of invalid intervals
*/

NPF_uint32_t  CurrentCESs;
/* C-bit Errored Seconds(CES) */
NPF_uint32_t  CurrentCSESs;
/* C-bit Severely Errored Seconds(CSES) */
NPF_uint32_t  CurrentCCVs;
/* C-bit Coding Violation (CCV) */
NPF_uint32_t  CurrentUASs;
/* Unavailable Seconds (UAS) */

} NPF_IfPDH_DS3_FECurStats_t;

/*
** DS3 Far End Interval Count/Statistics
*/
typedef struct {
/* 15 minute interval number - one of 96 during last 24 hours */

NPF_uint32_t         FarEndIntervalNumber ;
NPF_boolean_t        FarEndValidData;
NPF_IfPDH_DS3_FECurStats_t   FECurStats;

} NPF_IfPDH_DS3_FEIntervalStats_t;

/*
** DS3 Far End Total Count/Statistics
*/
typedef struct {

NPF_IfPDH_DS3_FECurStats_t  FETotalStats;

} NPF_IfPDH_DS3_FETotalStats_t;
```

```
/*
** DS3 Far End statistics
*/

typedef struct {

    NPF_IfPDH_DS3_FECurStats_t            CurStats;
    NPF_IfPDH_DS3_FEIntervalStats_t       IntervalStats;
    NPF_IfPDH_DS3_FETotalStats_t          TotalStats;

 } NPF_IfPDH_DS3_FarEndStats_t;


/*
** PDH Statistics
*/

struct NPF_IfPDH_Statistics {
      union{
   NPF_IfPDH_DS1_DS2_Stats_t  DS1_DS2Stats;
   NPF_IfPDH_DS3_Stats_t  DS3Stats;
      }u;
} ;



/*
 *    PDH Completion Callback Types
 */


/*
 *    PDH Completion Callback Types
 */
#define NPF_IF_PDH_LineSpeedAttrSet ((NPF_IF_TYPE_PDH<<16)+1)
#define NPF_IF_PDH_LineTypeAttrSet  ((NPF_IF_TYPE_PDH<<16)+2)
#define NPF_IF_PDH_LineCodingAttrSet ((NPF_IF_TYPE_PDH<<16)+3)
#define NPF_IF_PDH_SendCodeAttrSet ((NPF_IF_TYPE_PDH<<16)+4)
#define NPF_IF_PDH_LoopbackAttrSet ((NPF_IF_TYPE_PDH<<16)+5)
#define NPF_IF_PDH_SignalModeAttrSet ((NPF_IF_TYPE_PDH<<16)+6)
#define NPF_IF_PDH_TxClSrcAttrSet ((NPF_IF_TYPE_PDH<<16)+7)
#define NPF_IF_PDH_FDLinkAttrSet ((NPF_IF_TYPE_PDH<<16)+8)

         /* Fault Management */

#define NPF_IF_PDH_LineStatusQuery ((NPF_IF_TYPE_PDH<<16)+9)
#define NPF_IF_PDH_LoopbackStatusQuery ((NPF_IF_TYPE_PDH<<16)+10)

         /* Performance Monitoring */

#define NPF_IF_PDH_StatisticsQuery ((NPF_IF_TYPE_PDH<<16)+11)

/*
** End of Callback types for PDH
*/
```

```
#ifdef no_core_definition
/*
 *    Asynchronous response types for PDH interfaces
 *    To be included in the "npf_if_core.g"
 */


typedef struct NPF_IfPDH_LineStatus NPF_IfPDH_LineStatus_t   ;
typedef struct NPF_IfPDH_LoopbackStatus NPF_IfPDH_LoopbackStatus_t;
typedef struct NPF_IfPDH_Statistics NPF_IfPDH_Statistics_t;


       /* Fault Management Status */

       NPF_IfPDH_LineStatus_t        *if_LineStatus;
       NPF_IfPDH_LoopbackStatus_t    *if_LoopbackStatus;

       /* Performance Monitoring Statistics */

       NPF_IfPDH_Statistics_t        *if_PDHStats;
/*
** End of PDH async response type
*/
#endif /* no_core_definiton */

/*
** PDH Error Codes
*/

/*
** Macro
*/

#define NPF_IF_E_PDH_CODE(code) (0x10000+(NPF_IF_TYPE_PDH<<8)+(code))


/* Invalid PDH Interface Attribute */
#define NPF_IF_E_INVALID_PDH_ATTR NPF_IF_E_PDH_CODE(1)


/* Invalid PDH Interface Binding */
#define NPF_IF_E_INVALID_PDH_BINDING NPF_IF_E_PDH_CODE(2)


/* Invalid PDH Line Speed */
#define NPF_IF_E_INVALID_PDH_LINESPEED NPF_IF_E_PDH_CODE(3)

/* Invalid PDH Line Type */
#define NPF_IF_E_INVALID_PDH_LINETYPE NPF_IF_E_PDH_CODE(4)
/* Invalid PDH Line Coding */
#define NPF_IF_E_INVALID_PDH_LINECODING NPF_IF_E_PDH_CODE(5)

/* Invalid PDH Send Code */
#define NPF_IF_E_INVALID_PDH_SENDCODE NPF_IF_E_PDH_CODE(6)
```

```
/* Invalid PDH Loopback Configuration */
#define NPF_IF_E_INVALID_PDH_LOOPBACK NPF_IF_E_PDH_CODE(7)

/* Invalid PDH Signal Mode */
#define NPF_IF_E_INVALID_PDH_SIGNALMODE NPF_IF_E_PDH_CODE(8)

/* Invalid PDH Transmit Clock Source */
#define NPF_IF_E_INVALID_PDH NPF_IF_E_PDH_CODE(10)


/* Invalid PDH Facilities Data Link */
#define NPF_IF_E_INVALID_PDH_FDLINK NPF_IF_E_PDH_CODE(10)

/*
** End of PDH Error Codes
*/

/*
** PDH Interface Management Events
**
** These are extensions to existing Interface Management API events.
*/


/*
** PDH Interface Management Fault Management Events
*/

/*
 */



#define NPF_IF_PDH_BPV NPF_TYPE_PDH<<16) + 1)
           /* Bipolar Violation Error Event */
#define NPF_IF_PDH_EXZ NPF_TYPE_PDH<<16) + 2)
           /* Excessive Zeroes Error Event */
#define NPF_IF_PDH_LCV NPF_TYPE_PDH<<16) + 3)
           /* Line Coding Violation Event */
#define NPF_IF_PDH_PCV NPF_TYPE_PDH<<16) + 4)
           /* Path Coding Violation Event */
#define NPF_IF_PDH_CS NPF_TYPE_PDH<<16) + 5)
           /* Controlled Slip Error Event */
#define NPF_IF_PDH_OOF NPF_TYPE_PDH<<16) + 6)
           /* Out Of Frame */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 7)
           /* Alarm Indication Failure */


#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 8)
                                  /* RDI Failure (new name for RAI) */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 9)
                                  /* NCI Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 10)
                                  /* TIM Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 11)
                                  /* DEG Failure */
#define NPF_IF_PDH_AIS NPF_TYPE_PDH<<16) + 12)
```

```
                                    /* UNEQ Failure */



#define NPF_IF_PDH_LOF NPF_TYPE_PDH<<16) + 13)
           /* PDH Interface Loss Of Frame */
#define NPF_IF_PDH_LOS NPF_TYPE_PDH<<16) + 14)
           /* PDH Interface Loss Of Signal */
#define NPF_IF_PDH_LPF NPF_TYPE_PDH<<16) + 15)
           /* PDH Interface Loopback Pseudo-Failure */
#define NPF_IF_PDH_LOMF NPF_TYPE_PDH<<16) + 16)
           /* PDH Interface Loss of Multi-Frame Failure */
#define NPF_IF_PDH_RAI NPF_TYPE_PDH<<16) + 17)
           /* PDH Interface Remote Alarm Indication */



/*
**  PDH Performance Monitoring Events
*/


#define NPF_IF_PDH_ES NPF_TYPE_PDH<<16) + 100 + 1)
           /* Errored Seconds */
#define NPF_IF_PDH_SES NPF_TYPE_PDH<<16) + 100 + 2)
           /* Severe Errored Seconds */
#define NPF_IF_PDH_SEFS NPF_TYPE_PDH<<16) + 100 + 3)
/* Severely Errored Framing Seconds */
#define NPF_IF_PDH_CV NPF_TYPE_PDH<<16) + 100 + 4)
           /* Coding Violation */
#define NPF_IF_PDH_US NPF_TYPE_PDH<<16) + 100 + 5)
/* Unavailable Seconds */




#ifdef no_core_definition
/*
* PDH Event notification structure and array
* Defined in "npf_if_core.h"
*/
typedef struct NPF_IfEventData
{
NPF_IfEvent_t eventType;       /* Event type */
NPF_IfHandle_t handle;         /* Interface handle */
} NPF_IfEventData_t;


typedef struct {
NPF_uint16_t n_data;           /* Number of events in array */
NPF_IfEventData_t *eventData; /* Array of event notifications */
} NPF_IfEventArray_t;

typedef NPF_uint32_t NPF_IfEventHandlerHandle_t;

#endif /* EventData defined in the 'npf_if_core.h" */
```

```
/*
** Function calls
*/


/*
** NPF_IfPDH_LineStatusQuery
*/

NPF_error_t  NPF_IfPDH_LineStatusQuery(
NPF_IN NPF_callbackHandle_t    if_cbHandle,
NPF_IN NPF_correlator_t        if_cbCorrelator,
NPF_IN NPF_errorReporting_t    if_errorReporting,
NPF_IN NPF_IfHandle_t          if_Handle);



/*
** NPF_IfPDH_LoopbackStatusQuery
*/
NPF_error_t  NPF_IfPDH_LineStatusQuery(
NPF_IN NPF_callbackHandle_t    if_cbHandle,
NPF_IN NPF_correlator_t        if_cbCorrelator,
NPF_IN NPF_errorReporting_t    if_errorReporting,
NPF_IN NPF_IfHandle_t          if_Handle);

/*
** NPF_IfPDH_StatisticsQuery
*/
NPF_error_t  NPF_IfPDH_StatisticsQuery(
NPF_IN NPF_callbackHandle_t    if_cbHandle,
NPF_IN NPF_correlator_t        if_cbCorrelator,
NPF_IN NPF_errorReporting_t    if_errorReporting,
NPF_IN NPF_uint32_t            n_handles,
NPF_IN NPF_IfHandle_t         *if_HandleArray);

#endif
```

APPENDIX B LIST OF COMPANIES BELONGING TO NPF DURING APPROVAL PROCESS

Agere Systems
Altera
AMCC
Analog Devices
Avici Systems
Cypress Semiconductor
Enigma Semiconductor
Ericsson
Erlang Technologies
ETRI
EZ Chip
Flextronics
FutureSoft
HCL Technologies
Hi/fn
IBM
IDT
Intel
IP Fabrics
IP Infusion
Kawasaki LSI
Modular Networks
Motorola
NetLogic
Nokia
Nortel Networks
NTT Electronics
PMC Sierra
Seaway Networks
Sensory Networks
Sun Microsystems
TranSwitch
U4EA Group
Xelerated
Xilinx

APPENDIX C ACKNOWLEDGEMENTS

Working Group Chair:

Alex Conta, Transwitch, aconta@txc.com

Task Group Chair:

Alex Conta, Transwitch, aconta@txc.com

Task Group Editor:

John Renwick, Agere Systems, jrenwick@agere.com

The following individuals are acknowledged for their participation to IM API TG teleconferences, plenary meetings, mailing list, and/or for their NPF contributions used for the development of this Implementation Agreement. This list may not be all-inclusive since only names supplied by member companies for inclusion here will be listed. The NPF wishes to thank all active participants to this Implementation Agreement, whether listed here or not.

The list is in alphabetical order of last names:

Alex Conta (Transwitch)
Joe Esposito (Transwitch)
Huub Van Helvoort (Transwitch)
Bert Klaps (Transwitch)
Vinoj Kumar (Agere Systems)
Karen Nielsen (Ericsson)
Erik Pedersen (Ericsson)
John Renwick (Agere Systems)
Shatki Singh, (Transwitch)