Interface Management API Implementation Agreement (SONET/SDH Interfaces)

Revision 3.0

Editor: Tim Shanley, TranSwitch, tim.shanley@txc.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 Content

Table of Figures

Table of Tables

1 Introduction

A network element, for instance a switch or a router, has one or more physical connection points usually called *links,* through which it is connected to other network elements. Packets are received over a link by the network element for processing. A link usually has a physical medium for transmitting information, or Layer 1 (L1 – ISO Architecture Layered Model), and an associated information framing protocol, or Layer 2 (L2) protocol that is used to transfer certain structures of bits that is frames or packets, over the media of the link. SONET/SDH is a combination of such L1 and L2 mechanisms and protocols. In fact, the SONET/SDH capabilities are mapped into a multilayered hierarchy as described further in this section.

1.1 SONET/SDH Multilayered Hierarchy

"Physical Medium Layer" – it is responsible for encoding information in electrical or optical signals. This is Layer 1 (L1). It is called in this document *Physical Medium* or *Medium*.

"SONET Section Layer" or "SDH Regenerator Section Layer" – is responsible for the transmission of basic SONET/SDH frames; Transmission functions include framing, scrambling, and error monitoring. It is the basic building block and represents a single run of a cable or fiber between a pair of transmitter/receiver. The SONET *Section Layer* corresponds to the SDH *Regenerator Section Layer.* This is illustrated in Figure 1-1

"SONET Line Layer" or SDH Multiplexer Section Layer" – it is responsible for synchronization, multiplexing of data onto the SONET/SDH frames and protection switching. It is a transmission medium, which together with the associated equipment provides the transporting of information between two consecutive SONET/SDH network elements. The SONET *Line Layer* corresponds to the SDH *Multiplexing Section Layer.* This is illustrated in Figure 1-1.

"SONET or SDH Path Layer" – is responsible for end-to-end delivery of data or payloads at the appropriate speed. There are two subdivisions:

- SONET STS Path - corresponds to the SDH High Order Path. This sublayer delivers payloads or data at higher bandwidths, and speeds.

- SONET VT Path - corresponds to the SDH Lower Order Path. This sublayer delivers payloads or data at lower bandwidth, and speeds.

Both the SONET STS and SONET VT Path layers, respectively SDH High Order and SDH Low Order Path layers are using an adaptation sublayer to the SONET Line or SDH Multiplexing Section layer. The "mapping layer" maps user payloads into SONET STS SPE or SONET VT SPE or user data stored as containers into SDH Virtual Containers (SDH VC). The SONET STS or SDH High Order "adaptation sublayer" provides the adaptation from the SONET STS SPE or SDH High Order mapping layer, to the high order multiplexing layer. The SONET VT or SDH Low Order "adaptation sublayer" provides an adaptation of the low order mapping to the low order multiplexing. The output from low order multiplexing is further mapped into a SONET SPE, or high order SDH VC, and then further adapted, and multiplexed.

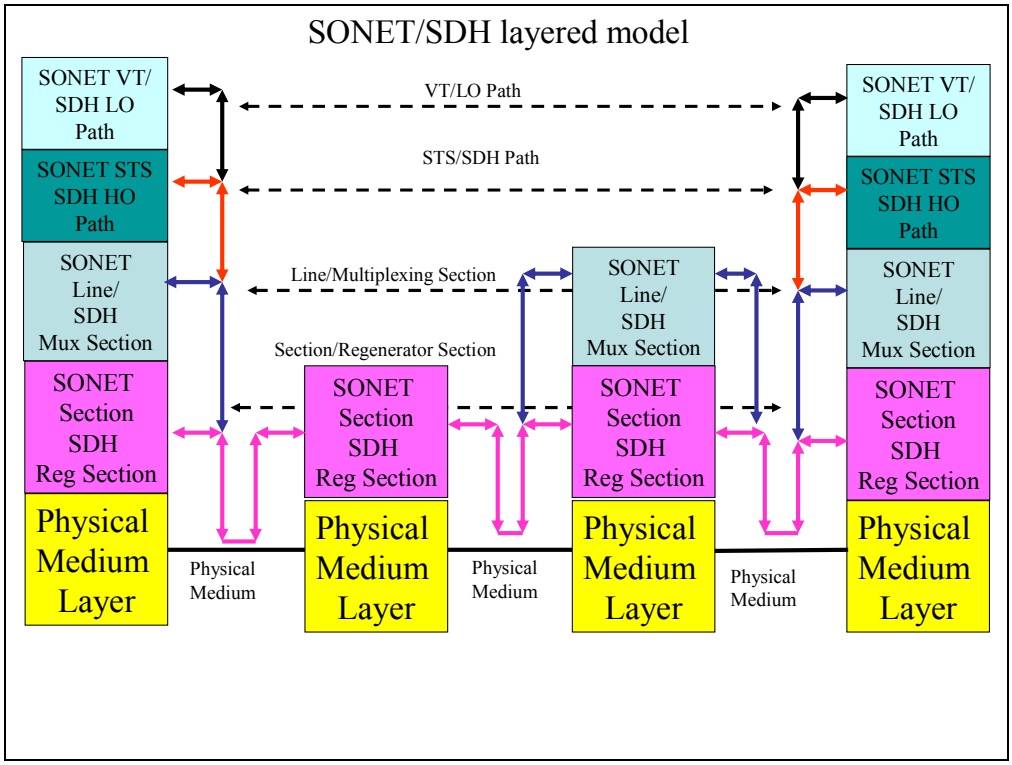

Figure 1-1 SONET/SDH Network Layering

Corresponding to these hierarchies, there is a hierarchy of terminating equipment:

- STE – Section Terminating Equipment (SONET) is a Network Element where a Section originates or terminates. It corresponds to the SDH Regenerator Section Terminating Equipment.

- LTE – Line Terminating Equipment (SONET) is a Network Element where a transport (section and line) overhead originates, is accessed, or terminates. It corresponds to the SDH Multiplexing Section Terminating Equipment.

- STS PTE – STS Path Terminating Equipment or a Network Element that multiplex/demultiplex STS payloads. A STS path corresponds to the SDH High Order Path. A VT Path corresponds to the SDH Lower Order Path.

An abstraction of a typical SONET/SDH network is:Figure 1-2.

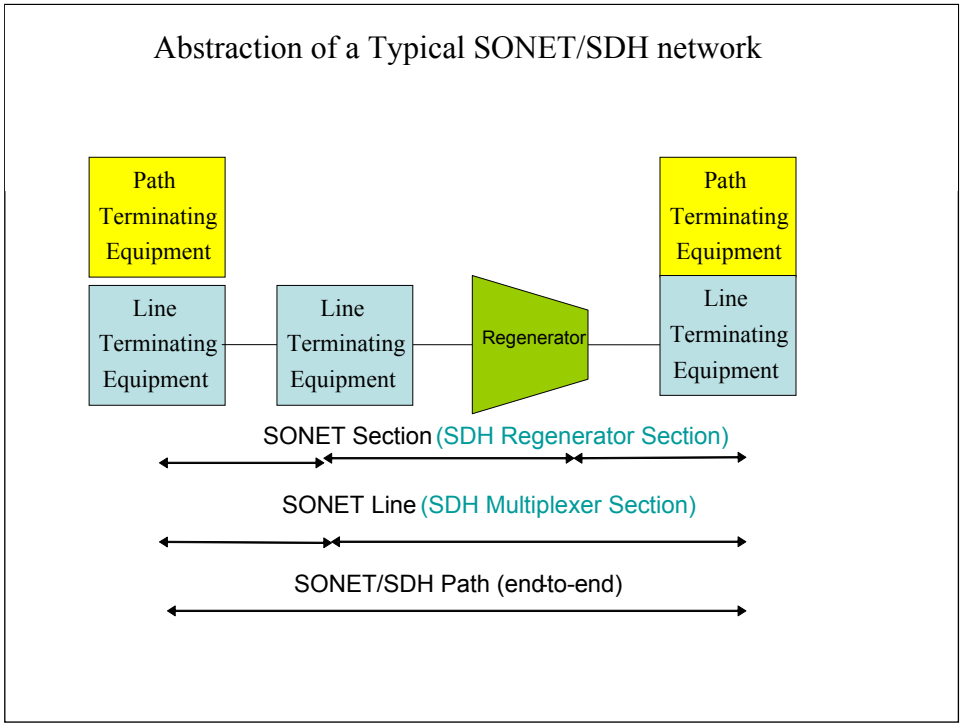

Figure 1-2 Abstraction of Typical SONET/SDH Network

1.1.1 SONET/SDH Concatenation

One important feature of SONET/SDH networks is the ability to sum bandwidth of smaller containers into a larger bandwidth container. This is a procedure whereby a multiplicity of Virtual Containers is associated one with another with the result that their combined capacity can be used as a single container across which bit sequence integrity is maintained. Two variants are used:

- Contiguous Concatenation

Frames resulted from Contiguous Concatenation have a suffix of Xc, where X is a number, for instance for VC-2 from 1, to 7.

- Virtual Concatenation

This is the SONET/SDH implementation of inverse multiplexing.
Frames resulted from Virtual Concatenation have a suffix of Xv, where X is a number, for instance for VC-2 from 1, to 64. For VC-3/4, X = 1,2, … 256.

1.1.2 SONET/SDH Protection

Two types of SONET/SDH protection are in use:

1.1.2.1 SONET/SDH Trail Protection

A SONET/SDH trail is a path between two termination points. Trail protection consists in having at least two trails: a working trail and a protecting trail. When the working trail fails or underperforms, it is replaced by the protecting trail. Several trail protecting schemes can be used

- 1+1 Protection

In a 1+1 SDH multiplex section protection system, two multiplex sections are provided; one carries the traffic, the other acts as a standby. A description of multiplex section 1+1 protection is given in ITU-T Recommendation G.783.

- 1:N Protection

A 1:N SDH multiplex section protection system consists of N traffic carrying multiplex sections that are to be protected, together with an additional multiplex section to provide protection. When not required for protection, this additional multiplex section capacity can be used to support lower priority "extra traffic". This extra traffic is not itself protected. A description of multiplex section 1:N protection together with the APS protocol is given in ITU-T Recommendation G.783.

- multiplex section shared protection rings

Multiplex section shared protection rings are characterized by dividing the total payload per multiplex section equally into working and protection capacity. The protection capacity is shared between multiple multiplex sections. A description of multiplex section shared protection rings, including the definition of the APS protocol, is provided in ITU-T Recommendation G.841.

- multiplex section dedicated rings

A multiplex section dedicated protection ring is a 1:N protection scheme where N = 1. A system consists of two counter rotating rings (each transmitting in opposite directions relative to each other). Under failure conditions, the entire working channel is looped to the protection channel.

1.1.2.2 SONET/SDH Subnetwork Protection

Subnetwork protection is described in ITU-T Recommendation G.805. It may be applied to either a SONET STS/VT, respectively SDH higher-order path, or lower-order path. To support subnetwork protection, two dedicated subnetwork connections are provided; one carries the traffic, the other acts as a standby. This protection mechanism can be used on any physical transport structure (e.g. meshed, rings or mixed). It can be used to protect a complete end-to-end network connection or a portion of a network connection. Further details of the application of this scheme in the SDH are provided in ITU-T Recommendation G.841.

1.2 SONET/SDH Hierarchy Abstraction in Software.

One of the major requirements for implementing software and software APIs for controlling/managing network devices is that the software can abstract and represent accurately the network architecture, the network layering, and/or the network hierarchy.

To better abstract and represent the SONET/SDH network hierarchy, this API provides a set of data structures, corresponding to the SONET/SDH layers and sublayers (see Figure 1-3).

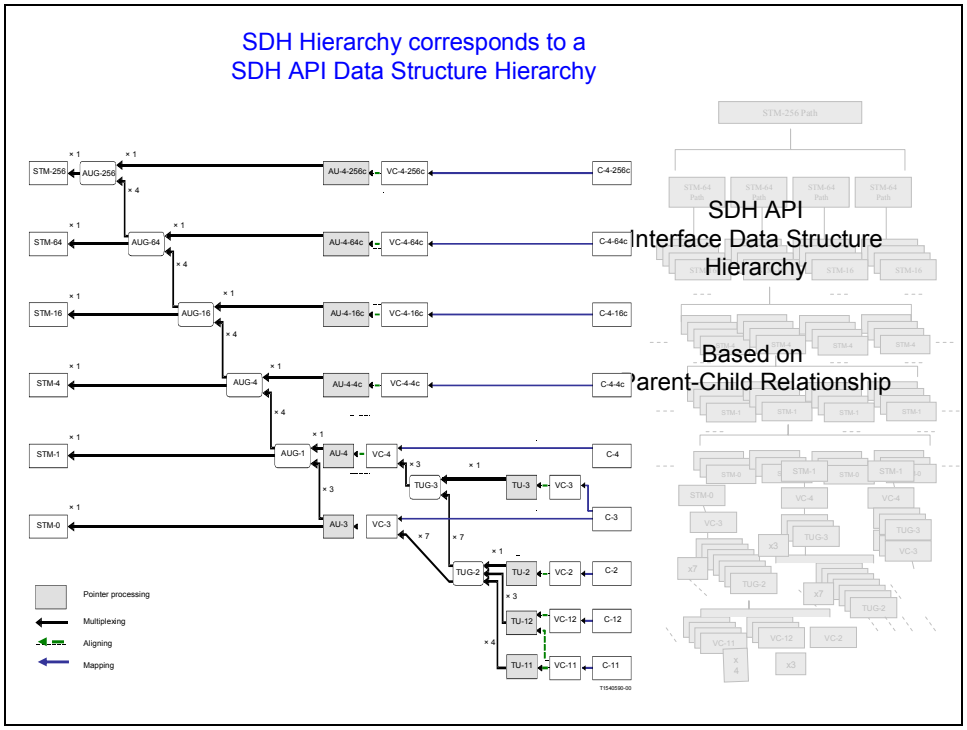

Figure 1-3 SONET/SDH Hierarchies and Software Data Structures relationships

For alignment with NPF Interface Management Implementations Agreement (IA), and NPF Software Framework IA, these SONET/SDH data structures are called “interfaces”. The SONET/SDH interfaces are well integrated with the “generic” and “specific interface type” data structures specified in the NPF Interface Management Implementation Agreement. They provide the fundamental building blocks for abstracting and representing the SONET/SDH network functions, and the framework around which the API Function Calls are organized, and grouped.

A SONET/SDH interface is associated with each element in the SONET/SDH hierarchy described above. The following type of interfaces are defined (see Figure 1-4):

1. Medium type interface – it abstracts and represents the “SONET/SDH physical medium layer”,
2. SONET Section, SDH Regenerator Section type interface – it abstracts and represents the “SONET Section layer”, or the “SDH Regeneration Layer”. A Section Termination Equipment (STE) would typically have a number of such interfaces.
3. SONET Line, SDH Multiplexing Section interface – it abstracts and represents the “SONET Line Layer” or SDH Multiplexing Section layer” at line level. Short Mapping Interface. A Line Termination Equipment (LTE) would typically have a number of such interfaces.
4. Adaptation interface – it abstracts and represents the “SONET or SDH adaptation sublayer” for adapting Mapping to Multiplexing. Although only a sublayer, adaptation requires a separate data structure because it may be performed by a different device than the one on which the mapping is performed, and because there can be a one to many correspondence between multiplexing and adaptation
5. SONET STS and VT Path, SDH High Order, LOW Order Path interfaces. Short Mapping interface – it abstracts and represents the “SONET or SDH mapping layer”. A Path Termination Equipment (PTE) would typically have a number of such interfaces.

Following the NPF Interface Management IA, the relationships between these interfaces are of a “parent-child’” type. These relationships can be established or removed by an application by invoking the NPF Interface Management API “Generic Interface Bind/Unbind function calls”.

These “parent-child” relationships allow the establishing of a data structure hierarchy, which corresponds almost one to one to the SONET/SDH hierarchy. Parent-child relationship is illustrated from left to right in Figure 1-4:

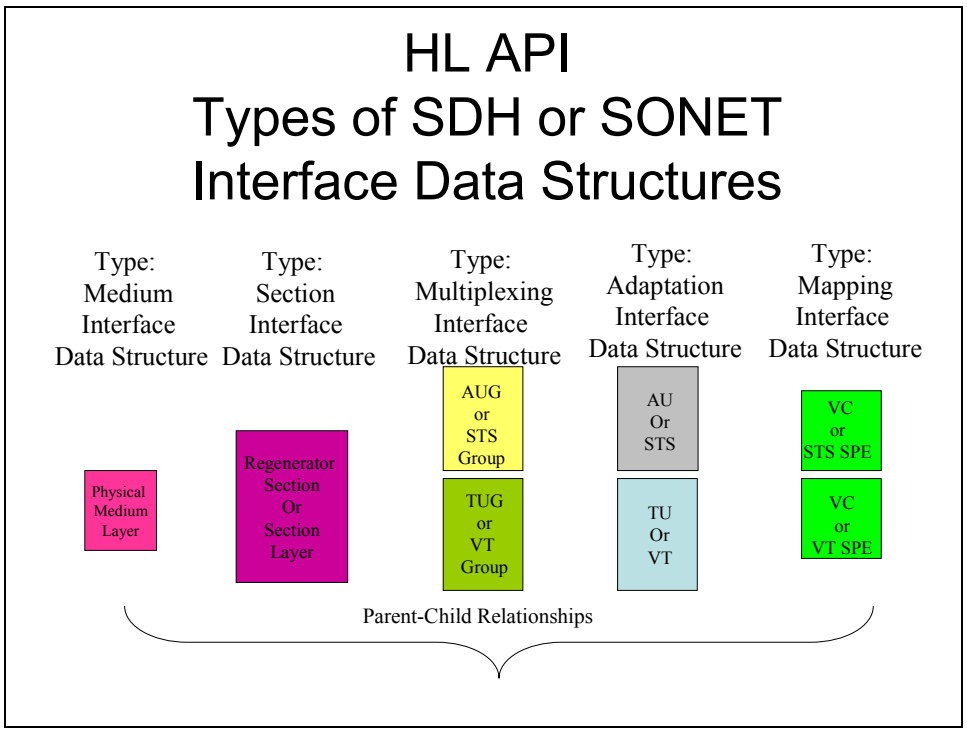

Figure 1-4 SONET/SDH Interface Types

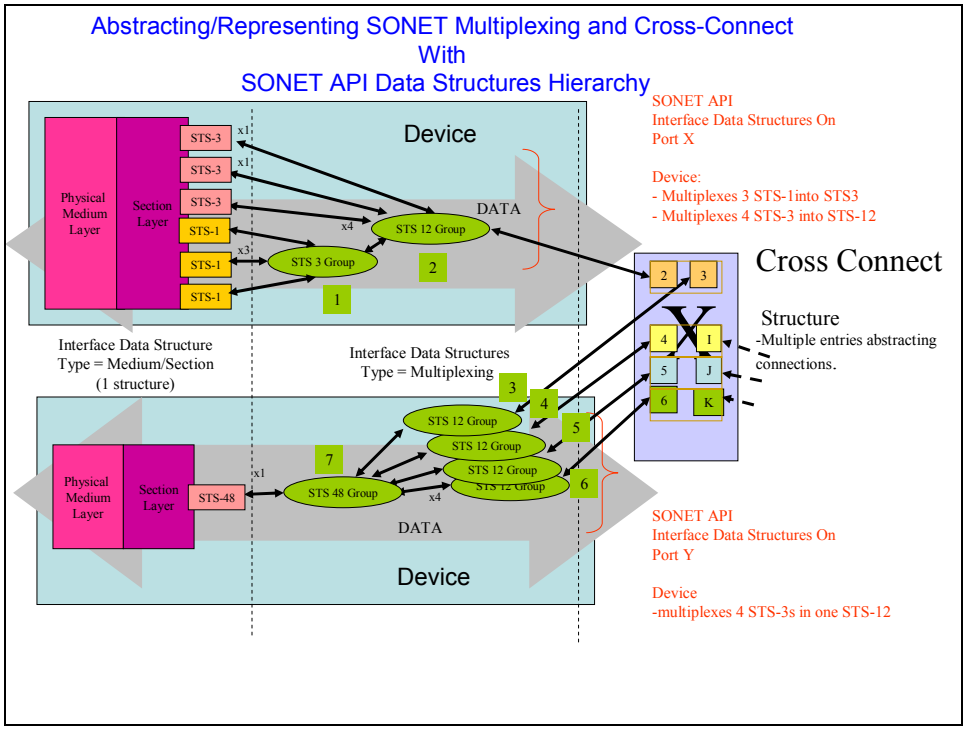

Figure 1-5 Using SONET/SDH data structures for SONET/SDH multiplexing

The illustration of two scenarios of abstracting the SONET/SDH network with NP Interface Management data structures, with parent-child relationships between them, is in Figure 1-5, and Figure 1-6 in SONET/SDH devices.

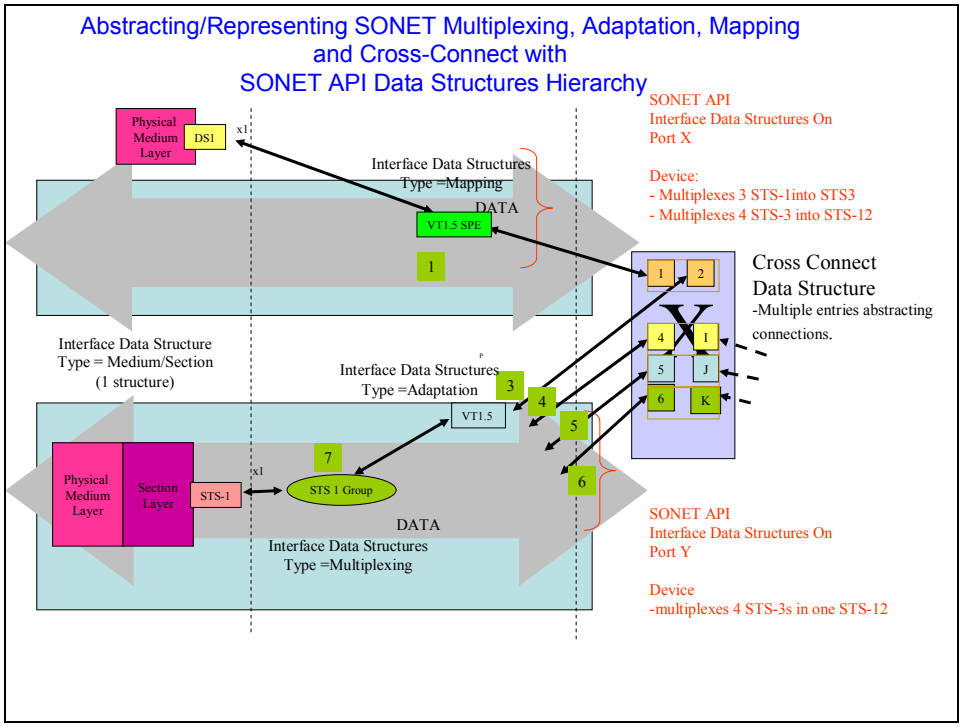

Figure 1-6 Using data structures in a SONET/SDH Network Element

1.3 SONET/SDH Functional Areas Targeted by the API.

There are several areas that the SONET/SDH Interface Management is addressing:

1.3.1 SONET/SDH Network Element General Configuration

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

1.3.2 SONET/SDH Fault Management - Alarm/Status Monitoring

Fault Management (FM) - Alarm/Status Monitoring - is a process that tracks failure events to contribute to a better understanding of the overall transmission performance of a SONET/SDH network element, and/or the components of a SONET/SDH network element (NE) (see Figure 1-7).

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.

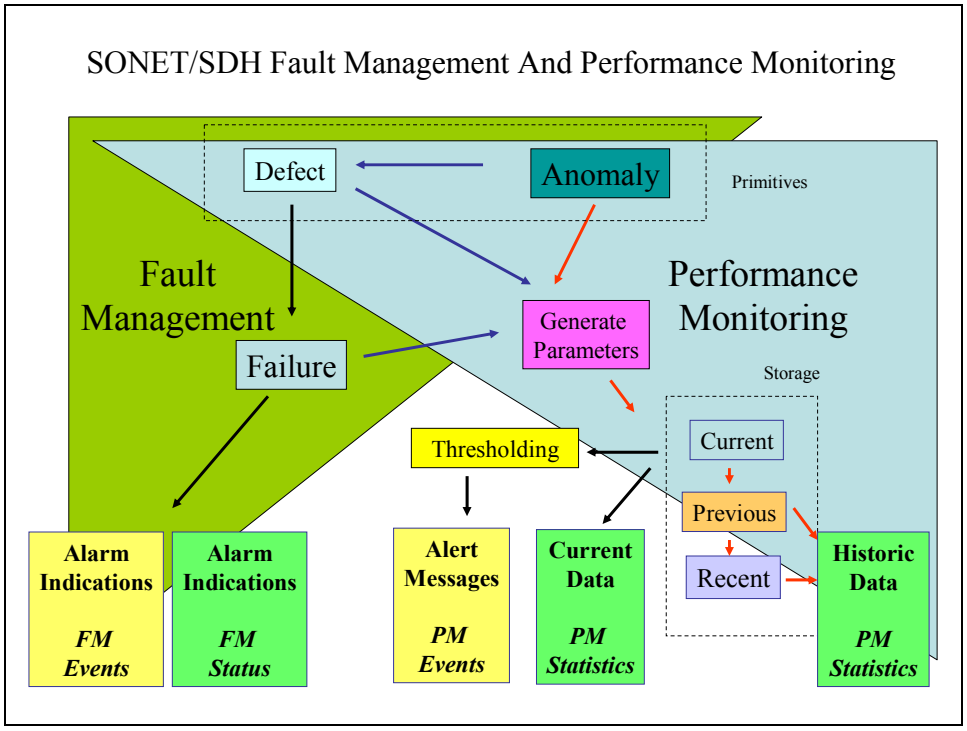

Figure 1-7 Fault Management and Performance Monitoring

1.3.3 SONET/SDH 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 gather, store, threshold, and report performance data (see Figure 1-7).

1.4 SONET/SDH as a Link Layer

A SONET/SDH network is used often as a link. Packet over SONET/SDH, ATM over SONET/SDH are examples of using SONET/SDH as a link. In such a use, the upper layer interface, for instance the ATM interface, has to be in a parent-child relationship

with the higher layer SONET/SDH interface, which is the SONET/SDH Mapping interface. Depending on the bandwidth used, this can be a SONET VT SPE Path Interface, respectively SDH Low Order Mapping (Path) interface, or a SONET STS SPE Path interface, respectively SDH High Order Mapping (Path) interface.

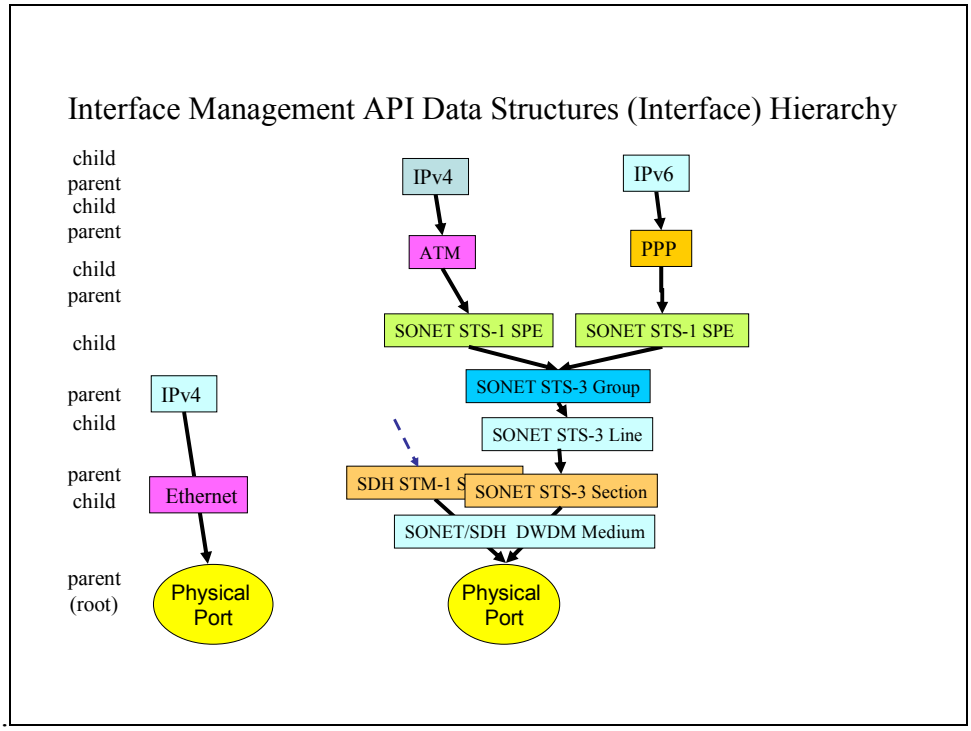

Figure 1-8 SONET/SDH interfaces as link layer interfaces

1.5 Assumptions and External Requirements

In the scope of this API, the structure and attributes of interfaces reflect what is needed by the data plane, or forwarding plane, not by the application software running in the control plane on a control processor. Such applications are expected to maintain their own representations of interfaces, very likely in a system that permits more and different interface attributes than are recognized by this API.

Memory allocation and usage model for the API implementation will be as dictated by the NPF Software Conventions Implementation Agreement

The API does not determine any policy with respect to operations on interfaces or their events. It is assumed that policy will be embodied in an Interface Manager module that is part of the application.

This SONET/SDH specific API assumes the existence of the Interface Management API Core Function Set, and shares all the same assumptions, and external requirements of that API.

1.6 Scope

The "SONET/SDH interfaces" addressed by this API are those related to external SONET, and/or SDH network ports only. External ports are ports that connect a network element to the network.

1.7 Dependencies

This document shares the same dependencies as the Interface Management API Core Function Set.

1.8 Document Organization

The document is organized in several sections:

Section 2 defines the SONET/SDH Data Structures. It is divided in subsections, based on the functions performed by the components of the data structures: the NPF interface structure, general interface attributes and configuration, fault management configuration, and status, performance monitoring, extensions to the NPF callback structure, NPF Error codes, and lastly SONET/SDH events.

Section 3 defines the SONET/SDH Function Calls. It is divided in subsections, based on roles played by the function calls: function calls to setup the software runtime environment: enable callbacks, and events, function calls to create data structures and set configuration parameters, function calls to configure and query Fault Management, function calls to configure and query Performance Monitoring.

Section 4 is a summary of the SONET/SDH Function Calls.

Section 5 is a reference section.

Section 6 is Revision History

Secion7 is an Appendix containing the .h file with all SONET/SDH API data structure and function call definitions.

1.9 Document Terminology

API stands for Application Programming Interface, which in software has the meaning of a set of data structures and function calls (or functions, or calls). This note is to avoid confusion with the use of the term "API" as to mean "function call".

2 SONET/SDH NPF Interface Management Data Types

2.1 SONET/SDH NPF Interface Data Structures

This section describes the SONET/SDH Interface data structures, as well as extensions to the Generic Interface Data structure related to SONET/SDH support. The following data SONET/SDH interface types are defined:

- SONET/SDH Physical Medium interface,
- SONET Section, SDH Regeneration Section interface,
- SONET Line, SDH Multiplexing Section interface,
- SONET/SDH Adaptation interface,
- SONET/SDH Path (Mapping) interface,

These data structures have a parent-child relationship, in which the root is the SONET/SDH Physical Medium interface.

Note:
Two SONET/SDH mechanisms are also abstracted through some of the SONET/SDH Interfaces:

- SONET/SDH Virtual Concatenation - the Mapping interfaces
- SONET/SDH Automatic Protection Scheme (APS) – the Multiplexing and Mapping interfaces.

2.1.1 SONET/SDH Interface Type

SONET/SDH interfaces are part of the SONET/SDH interface type.

For supporting the SONET/SDH interface, the SONET/SDH type must be added to the Core Interface Management API interface structure.

```
#define NPF_IF_TYPE_SONET_SDH   10    /* SONET-SDH Interface */
```

2.1.2 SONET/SDH NPF Interface Attributes Data Structures

A forward reference to the SONET/SDH NPF Interface Attributes data structure must be added to the **NPF_IfGeneric_t** structure in **npf_if_core.h** if SONET/SDH interfaces are supported. So, before the declaration of **NPF_IfGeneric_t**, the following must appear:

```
typedef struct NPF_IfSONET_SDH NPF_IfSONET_SDH_t;
```

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

```
        NPF_IfSONET_SDH_t *SONET_SDH_Attr; /* SONET_SDH interface attributes */
```

```
/*
**  SONET-SDH Interface Attributes
*/
struct NPF_IfSONET_SDH {

       NPF_IfSONET_SDH_Type_t   SONET_SDH_IfType; /* SONET-SDH Interface type */

       union {

              NPF_IfSONET_SDH_Medium_t SONET_SDH_Medium;

                     /* SONET-SDH Physical Medium Interface Attributes */

              NPF_IfSONET_SDH_Section_t SONET_SDH_Section;

                     /* SONET Section SDH Regenerator Section Interface Attributes */

              NPF_IfSONET_SDH_Multiplexing_t SONET_SDH_Multiplexing;

                     /* SONET-SDH Multiplexing Attributes */

              NPF_IfSONET_SDH_Adaptation_t SONET_SDH_Adaptation;

                     /* SONET-SDH Adaptation Attributes */

              NPF_IfSONET_SDH_Mapping_t SONET_SDH_Mapping;

                     /* SONET-SDH Mapping Attributes */


        }u;
       };
```

2.1.2.1 SONET/SDH Interface Types

The following SONET/SDH interface types are defined:

```
typedef enum {

/*
** SONET-SDH  Interface Types
*/

NPF_IF_TYPE_SONET_SDH_Medium=1,       /* SONET-SDH  Physical Medium  */
NPF_IF_TYPE_SONET_SDH_Section=2,      /* SONET-SDH  Section Interface */
NPF_IF_TYPE_SONET_SDH_Mux =3,         /* SONET-SDH  Multiplexing interface */
NPF_IF_TYPE_SONET_SDH_Adaptation =4,  /* SONET-SDH  Adaptation interface */
NPF_IF_TYPE_SONET_SDH_Mapping=5       /* SONET-SDH  Mapping interface */
/*
```

```
** End of Interface types for SONET-SDH
*/

} NPF_IfSONET_SDH_Type_t;
```

2.1.2.2 SONET/SDH Physical Medium Interface Attributes

The SONET/SDH Physical Medium Interface, or shorter SONET/SDH Medium Interface is the root of a SONET/SDH interface hierarchy. It has parent-child relation with the Section interfaces. It is the parent of the SONET Section, or SDH Regenerator Section interfaces.

```
/*
** SONET-SDH  Physical Medium Interface Attributes
*/
typedef struct {
         NPF_uint32_t                port;                  /* Physical Port Number */

        NPF_IfSONET_SDH_MediumCoding_t  MediumLineCoding;    /* Medium Line Coding */
        NPF_IfSONET_SDH_MediumLineType_t MediumLineType;     /* Medium Line Type */
        NPF_IfSONET_SDH_MediumLoopbackConfig_t MediumLoopbackConfig ;
                                                    /* Medium Loop-back Configuration */
        NPF_IfSONET_SDH_MediumSESthresholdSet_t MediumSESthresholdSet ;
                                    /* Medium SES recognized set of thresholds */
        NPF_IfSONET_SDH_MediumStatus_t  MediumStatus;
                                                    /* Medium Status */
        NPF_IfSONET_SDH_MediumCurrStatus_t  MediumCurrentStatus;
                                                    /* Medium Current Status */

         } NPF_IfSONET_SDH_Medium_t;
```

The elements of the structures are further defined as:

2.1.2.2.1 SONET/SDH Medium Encoding Attribute

```
/*
**  SONET-SDH  Medium Encoding
*/
typedef enum {
  NPF_IF_SONET_SDH_MediumCoding_Other = 1,   /* Coding is other */
  NPF_IF_SONET_SDH_MediumCoding_B3ZS = 2,    /* Coding is B3ZS */
  NPF_IF_SONET_SDH_MediumCoding_CMI = 3,     /* Coding is CMI */
  NPF_IF_SONET_SDH_MediumCoding_NRZ = 4,     /* Coding is NRZ */
  NPF_IF_SONET_SDH_MediumCoding_RZ = 5       /* Coding is RZ */

/* tbd: optical types  */

} NPF_IfSONET_SDH_MediumCoding_t;
```

2.1.2.2.2 SONET/SDH Medium Line Type Attribute

```
/*
```

```
**  SONET-SDH  Medium Line Type
*/
typedef enum {
  NPF_IF_SONET_SDH_MediumLine_Other = 1,
                                    /* Medium Line type is Other */
  NPF_IF_SONET_SDH_MediumLine_ShortSingleMode = 2,
                                    /* Medium Line type is Short Single Mode */
  NPF_IF_SONET_SDH_MediumLine_LongSingleMode = 3,
                                    /* Medium Line type is Long Single Mode */
  NPF_IF_SONET_SDH_MediumLine_MultiMode = 4,
                                    /* Medium Line type is Multi Mode */
  NPF_IF_SONET_SDH_MediumLine_Coax = 5,
                                    /* Medium Line type is Coax */
  NPF_IF_SONET_SDH_MediumLine_UTP = 6
                                    /* Medium Line type is UTP */
} NPF_IfSONET_SDH_MediumLineType_t;
```

2.1.2.2.3 SONET/SDH Medium Loopback Configuration Attribute

```
/*
**  SONET-SDH  Medium Loopback Configuraton
*/
typedef enum {
  NPF_IF_SONET_SDH_NoLoop = 0,
                                    /* NO Loop */
  NPF_IF_SONET_SDH_FacilityLoop = 1,
                                    /* SONET-SDH Facility Loop */
  NPF_IF_SONET_SDH_TerminalLoop = 2,
                                    /* SONET-SDH Terminal Loop */
  NPF_IF_SONET_SDH_OtherLoop = 3
                                    /* SONET-SDH Other Loop */

} NPF_IfSONET_SDH_MediumLoopbackConfig_t;


/* Shakti : */
```

The elements of the *status structure*, as well as those of the *current status structure* are defined in the Fault Management Section (2.1.3.2.1).

2.1.2.3 SONET Section or SDH Regenerator Section Interface Attributes

The SONET Section or SDH Regenerator Section Interface is the second layer in the SONET/SDH interface hierarchy. It has a parent-child relationship with the Medium Interface, and the Multiplexing Interfaces. It is a child of a Medium interface, and it is the parent or one or more Multiplexing Interfaces.

```
/*
**  SONET Section or SDH Regenerator Section Interface Attributes
*/
typedef struct {

        NPF_IfSONET_SDH_SectionType_t  SectionType;
        union{
        NPF_IfSONET_SectionIf_t              SectionIf;
```

```
        NPF_IfSDH_RegSectionIf_t      RegSectionIf;
        }u;

        NPF_IfSONET_SDH_Section_FMConf_t      Section_FM_Conf;


        NPF_IfSONET_SDH_SectionStatus_t  SectionStatus ;

               } NPF_IfSONET_SDH_Section_t;

/*
**  end SONET Section or SDH Regenerator Section Interface
*/
```

2.1.2.3.1 SONET/SDH Section Type Attribute

```
/*
**  SONET-SDH  Section Type
*/

typedef enum {
  NPF_IF_SONETSDH_Section_SONET = 1,         /* Section is SONET */
  NPF_IF_SONETSDH_Section_SDH = 2            /* Section is SDH */
        } NPF_IfSONET_SDH_SectionType_t;
```

2.1.2.3.2 SDH Regenerator Section Interface Attributes

```
/*
** SDH Regenerator Section
*/

typedef enum {

  NPF_IF_SDH_RegSection_none = 0,     /* Type not defined */
  NPF_IF_SDH_RegSection_STM_0 = 1     /* STM-0 51.84Mbps */
  NPF_IF_SDH_RegSection_STM_1 = 2,    /* STM-1 155.55Mbps */
  NPF_IF_SDH_RegSection_STM_4 = 3,    /* STM-4 622.08Mbps */
  NPF_IF_SDH_RegSection_STM_16 = 4,   /* STM-16 2.48Gbps */
  NPF_IF_SDH_RegSection_STM_64 = 5,   /* STM-64 9.92Gbps */
  NPF_IF_SDH_RegSection_STM_256 = 6   /* STM-256 39.68Gbps */

} NPF_IfSDH_RegSectionIf_t;
```

2.1.2.3.3 SONET Section Interface Attributes

```
/*
** SONET Section
*/
typedef enum {

  NPF_IF_SONET_Section_none = 0,      /* Type not defined */
  NPF_IF_SONET_Section_STS_1 = 1      /* STS1 51.84Mbps */
  NPF_IF_SONET_Section_STS_3 = 2,     /* STS3 155.55Mbps */
  NPF_IF_SONET_Section_STS_12 = 3,    /* STS12 622.08Mbps */
  NPF_IF_SONET_Section_STS_48 = 4,    /* STS48 2.48Gbps */
  NPF_IF_SONET_Section_STS_192 = 5,   /* STS192 9.92Gbps */
  NPF_IF_SONET_Section_STS_768 = 6    /* STS768 39.68Gbps */

} NPF_IfSONET_SectionIf_t;
```

The elements of the configuration and status structures are defined in the Fault Management Section. The elements of the Performance Monitoring configuration and statistics structures are defined in the Performance Monitoring Section

2.1.2.4 SONET Line or SDH Multiplexing Section Interface Attributes

SONET or SDH multiplexing is a procedure by which multiple lower order path layer signals are adapted into a higher order path or the multiple higher order path layer signals are adapted into a SONET line or respectively SDH multiplex section.

The SONET/SDH Multiplexing Interface is abstracting the Multiplexing functions of SONET/SDH. It has a parent-child relationship with

A SONET/SDH High Order Multiplexing interface has a parent-child relationship with SONET Section or SDH Regenerator Section interfaces (layer below), and SONET/SDH Adaptation interfaces (layer above) - below is towards the network, above is towards the user of the network. The High Order SONET/SDH interfaces are children of the Section interfaces, and parents of the Adaptation interfaces.

A SONET Line, or SDH Low Order Multiplexing interface has a parent-child relationship with a SONET/SDH High Order Mapping interface (layer below), and Low Order SONET/SDH Adaptation interfaces (layer above) - below is towards the network, above is towards the user of the network. Low Order Multiplexing interfaces are children of the Higher Order Mapping interfaces, and parents of Low Order Adaptation interfaces.

There are two types: high order and low order multiplexing interfaces:

```
/*
**      SONET-SDH  Multiplexing Interface attributes,
**      used in the union within NPF_IfGeneric_t.
*/
typedef struct {


        union {

        NPF_IF_SDH_HO_MuxIf_t         SDH_HO_Multiplexing;
        NPF_IF_SDH_LO_MuxIf_t         SDH_LO_Multiplexing;
        NPF_IF_SONET_STS_MuxIf_t      SONET_STS_Multiplexing;
        NPF_IF_SONET_VT_MuxIf_t       SONET_VT_Multiplexing;

        } u;

        NPF_IfSONET_SDH_Protection_t  Mux_Protection;

        NPF_IfSONET_SDH_Mux_FMConf_t          Mux_FM_Conf;
        NPF_IfSONET_SDH_MuxStatus_t           MultiplexingStatus;

                } NPF_IfSONET_SDH_Multiplexing_t;
```

The elements of the configuration NPF_IfSONET_SDH_Mux_FMConf_t, and status structures NPF_IfSONET_SDH_MuxStatus_t are defined in the Fault Management Section (2.1.3.2.).

2.1.2.4.1 SDH High Order Multiplexing Interface Attributes

The High Order Multiplexing interface is abstracting the multiplexing of high order SDH Virtual Containers (VCs).

The following multiplexing is supported (see Figure 2-1):

- SDH
    - AUG-256 – Administrative Unit Group 256. A AUG-256 Multiplexing interface can abstract the multiplexing of:
        - One (1) AU-4-256c
        - Four (4) AUG-64

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) AU-4-256c adaptation interface
        - Four (4) AUG-64 multiplexing interfaces

    - AUG-64– Administrative Unit Group 64. A AUG-64 Multiplexing interface can abstract the multiplexing of:
        - One (1) AU-4-64c
        - Four (4) AUG-16

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) AU-4-4c adaptation interface
        - Four (4) AUG-16 multiplexing interfaces

    - AUG-16– Administrative Unit Group 16. A AUG-16 Multiplexing interface can abstract the multiplexing of:
        - One (1) AU-4-16c
        - Four (4) AUG-4

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) AU-4-16c adaptation interface
        - Four (4) AUG-4 multiplexing interfaces

    - AUG-4– Administrative Unit Group 4. A AUG-4 Multiplexing interface can abstract the multiplexing of:
        - One (1) AU-4-4c
        - Four (4) AUG-1

This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:

- One (1) AU-4-4c adaptation interface
- Four (4) AUG-1 multiplexing interfaces

- AUG-1 – Administrative Unit Group 1. A AUG-1 Multiplexing interface can abstract the multiplexing of:
  - One (1) AU-4
  - Three (3) AU-3

This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:

- One (1) AU-4 adaptation interface
- Three (4) AU-3 adaptation interfaces

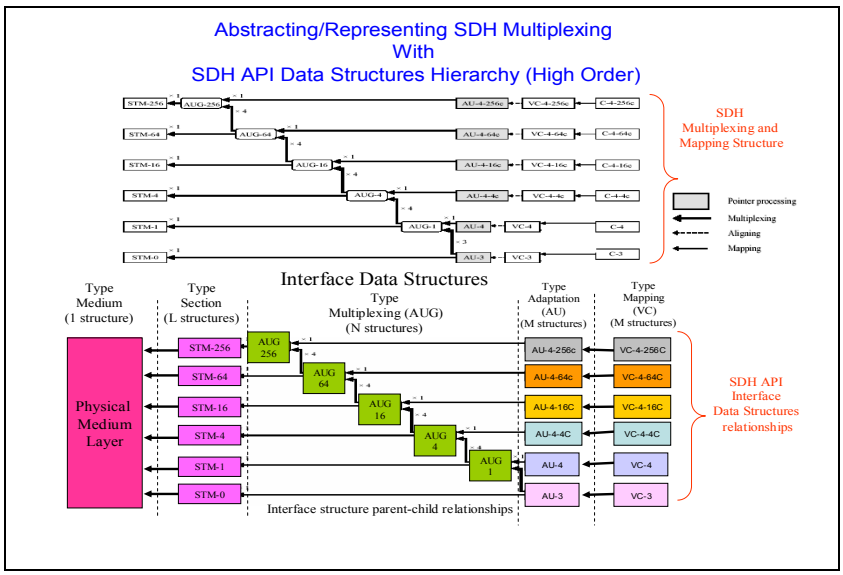

Figure 2-1 SDH High Order Multiplexing, Adaptation, and Mapping

The SDH High Order Multiplexing Attributes data structure is:

```
/*
** SDH High Order Multiplexing Interface Attributes
*/
typedef enum {

  NPF_IfSDH_Mux_none = 0,      /* Type of multiplexing not defined */
  NPF_IfSDH_Mux_AUG_1 = 1,     /* Multiplexing in AUG 1 */
  NPF_IfSDH_Mux_AUG_4 = 2,     /* Multiplexing in AUG 4 */
  NPF_IfSDH_Mux_AUG_16 = 3,    /* Multiplexing in AUG 16 */
  NPF_IfSDH_Mux_Aug_64 = 4,    /* Multiplexing in AUG 64 */
  NPF_IfSDH_Mux_Aug_256 = 5    /* Multiplexing in AUG 256 */

}NPF_IfDH_HO_MuxIf_t
```

2.1.2.4.2 SONET STS (High Order) Line Multiplexing Interface Attributes

The SONET STS Multiplexing interface is abstracting the multiplexing of high order SONET STS Synchronous Payload Envelopes (STS SPEs) .

The following multiplexing is supported (See Figure 2-2):

- SONET
    - STS-768 – Synchronous Transport Signal 768. A STS-768 Multiplexing interface can abstract the multiplexing of:
        - One (1) STS-768c
        - Four (4) STS-192

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) STS-768c adaptation interface
        - Four (4) STS-192 multiplexing interfaces

    - STS-192– Synchronous Transport Signal 192. A STS-192 Multiplexing interface can abstract the multiplexing of:
        - One (1) STS-192c
        - Four (4) STS-48

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) STS-192c adaptation interface
        - Four (4) STS-48 multiplexing interfaces

    - STS-48– Synchronous Transport Signal 48. A STS-48 Multiplexing interface can abstract the multiplexing of:
        - One (1) STS-48c
        - Four (4) STS-12

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) STS-48c adaptation interface
        - Four (4) STS-12 multiplexing interfaces

    - STS-12– Synchronous Transport Signal 12. A STS-12 Multiplexing interface can abstract the multiplexing of:
        - One (1) STS-12c
        - Four (4) STS-3

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:
        - One (1) STS-12c adaptation interface
        - Four (4) STS-3 multiplexing interfaces

- STS-3 – Synchronous Transport Signal 3. A STS-3 Multiplexing interface can abstract the multiplexing of:
  - One (1) STS-3c
  - Three (3) STS-1

This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:

  - One (1) STS-3c adaptation interface
  - Three (3) STS-1 multiplexing interfaces

- STS-1 – Synchronous Transport Signal 1. A STS-1 Multiplexing interface can abstract the multiplexing of:
  - One (1) STS-1
  - One (1) VT Group Multiplexing interface (See VT Multiplexing Interface subsection)

This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:

  - One (1) STS-1
  - One (1) VT Group Multiplexing interface (See VT Multiplexing Interface subsection)

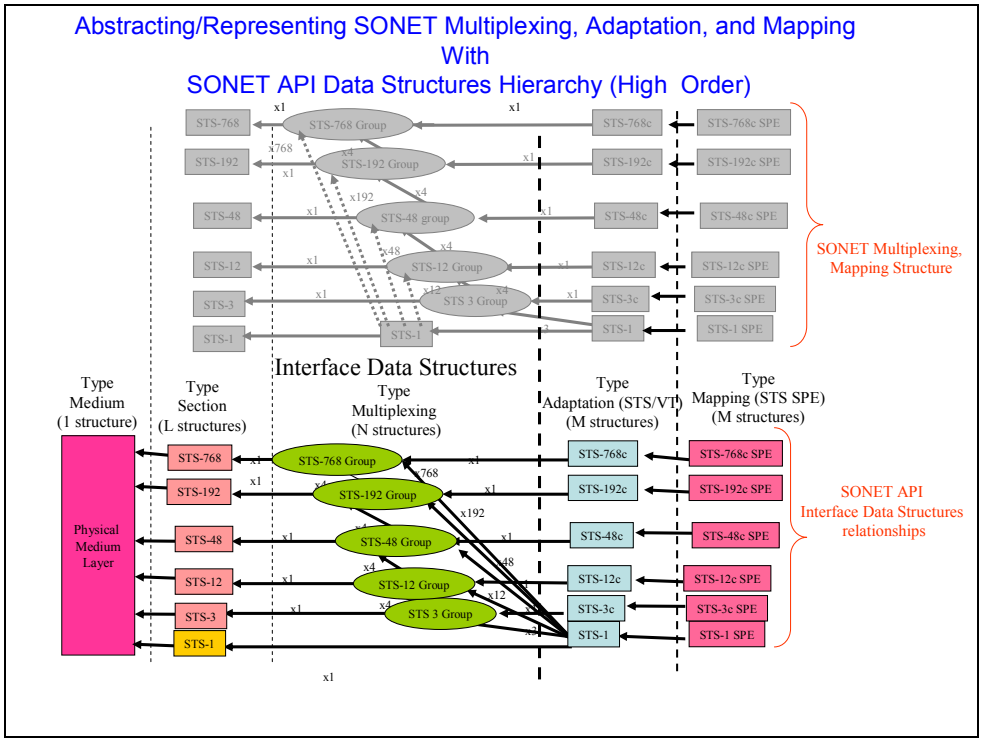

Figure 2-2 SONET STS Multiplexing, Adaptation and Mapping

The following data structure is defined:

```
/*
** SONET STS Multiplexing Interface Attributes
```

```
*/
typedef enum {
  NPF_IF_SONET_Mux_none = 0,  /* Type of multiplexing not defined */
  NPF_IF_SONET_Mux_STS_1_Group = 1,   /* Multiplexing into STS-1  */
  NPF_IF_SONET_Mux_STS_3_Group = 2,   /* Multiplexing into STS-3  */
  NPF_IF_SONET_Mux_STS_12_Group = 3,  /* Multiplexing into STS-12 */
  NPF_IF_SONET_Mux_STS_48_Group = 4,  /* Multiplexing into STS-48 */
  NPF_IF_SONET_Mux_STS_192_Group = 5, /* Multiplexing into STS-192 */
  NPF_IF_SONET_Mux_STS_768_Group = 6  /* Multiplexing into STS-768 */

}NPF_IfSONET_STS_MuxIf_t
```

2.1.2.4.3 SDH Low Order Multiplexing Interface Attributes

The Low Order Multiplexing interface is abstracting the multiplexing of low order SDH Virtual Containers (VCs).

The following multiplexing is supported (See Figure 2-3):

- SDH
    - TUG-3 - Tributary Unit Group 3. A TUG-3 Multiplexing Interface can abstract the multiplexing of
        - One (1) TU-3 or
        - Seven (7) TUG-2

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation or multiplexing interfaces as follows:

        - One (1) TU-3 adaptation interface, or
        - Seven (7) TUG-2 type tributary unit group multiplexing interfaces. This allows a two level multiplexing at low order level.

Note: A TUG-3 consists of a homogeneous assembly of TUG-2s or a TU-3.

    - TUG-2: Tributary Unit Group 2. A TUG-2 Multiplexing Interface can abstract the multiplexing of
        - One TU-2
        - Three (3) TU-12
        - Four (4) TU-11

      This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation interfaces, as follows:

        - One TU-2 adaptation interface
        - Three (3) TU-12 adaptation interfaces
        - Four (4) TU-11 adaptation interfaces

Note: A TUG-2 consists of a homogeneous assembly of identical TU-11s and TU-12s, or a TU-2.

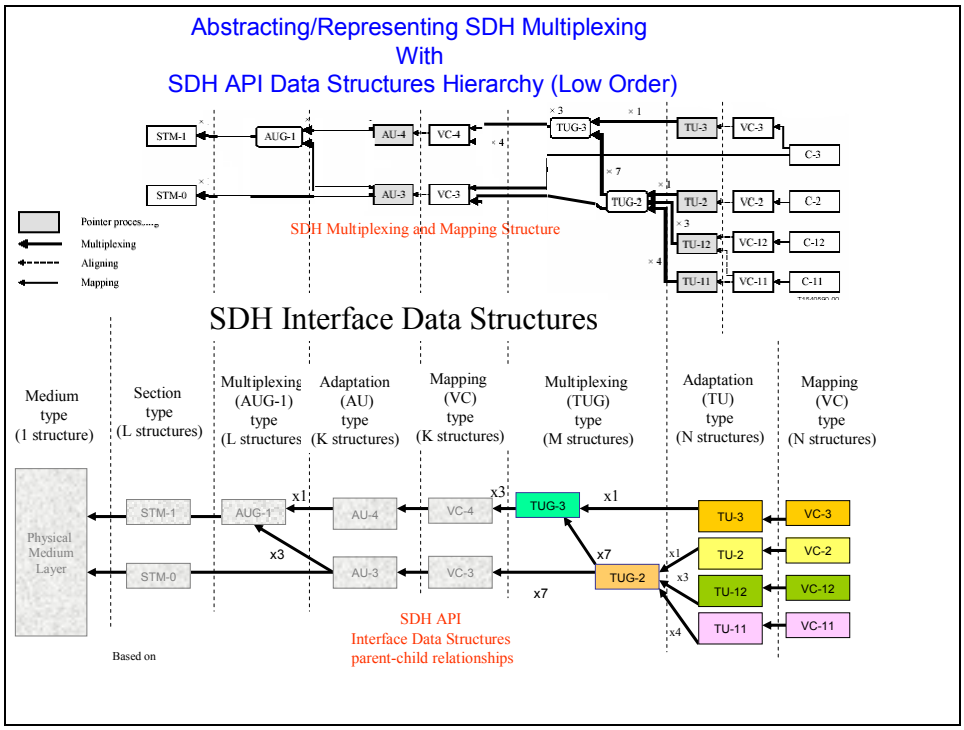

Figure 2-3 SDH Low Order Multiplexing, Adaptation and Mapping

The following data structure is defined:

```
/*
** SDH Low Order Multiplexing Interface Attributes
*/
typedef enum {

  NPF_IF_SDH_Mux_TUG_2 = 6,    /* Multiplexing in TUG 2 */
  NPF_IF_SDH_Mux_TUG_3 = 7     /* Multiplexing in TUG 3 */

} NPF_IfSDH_LO_MuxIf_t;
```

2.1.2.4.4 SONET VT (Low Order) Multiplexing Interface Attributes

The Low Order Multiplexing interface is abstracting the multiplexing of low order SONET Virtual Tributaries Synchronous Payload Envelopes (VT SPEs) (see Figure 2-4).

The following multiplexing is supported:

- SONET
  - VT Group - Virtual Tributary Group. A VT Group multiplexing interface can abstract and represent the multiplexing of:

- One (1) VT6
- Two (2) VT3
- Three (3) VT2
- Four (4) VT1.5

This abstracting is materialized by establishing parent-child relationships with the corresponding adaptation interfaces, as follows:

- One (1) VT6 adaptation interface
- Two (2) VT3 adaptation interfaces
- Three (3) VT2 adaptation interfaces
- Four (4) VT1.5 adaptation interfaces

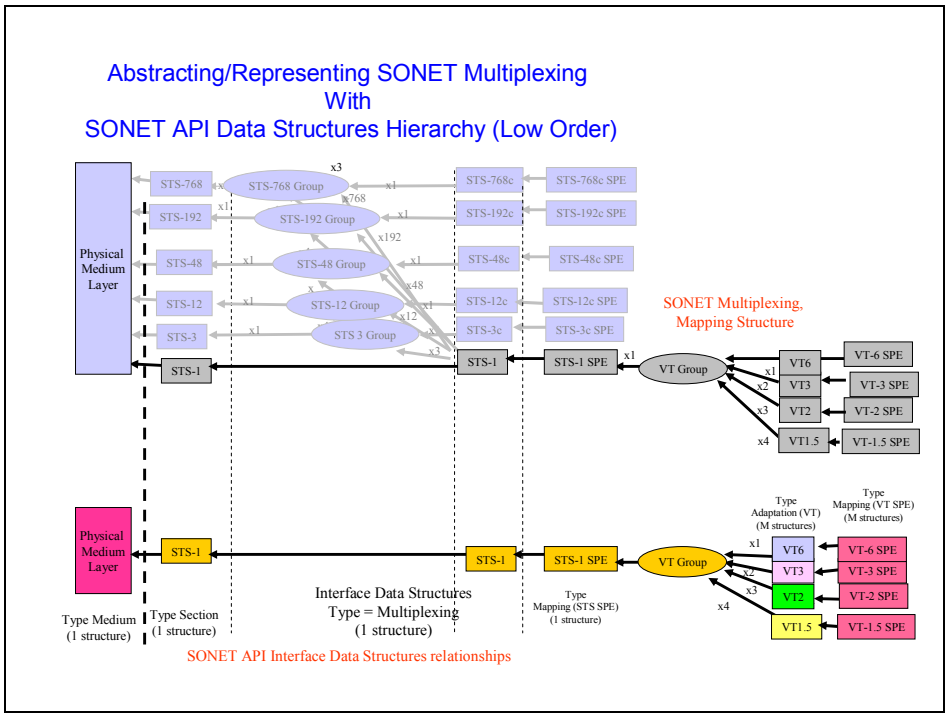

Figure 2-4 SONET VT Multiplexing, Adaptation, and Mapping

The following data structures are defined:

```
/*
** SONET VT Group Multiplexing Interface Attributes
*/
typedef enum {

  NPF_IF_SONET_Mux_VT_Group = 7       /* Multiplexing into VT Group */

}NPF_IfSONET_VT_MuxIf_t
```

2.1.2.4.5 SONET Line, SDH Multiplexing Section Protection

The SONET Line, or SDH Multiplexing Section Protection is abstracted by one additional level of interfaces (Figure 2-5.)

The interfaces on this level are still Multiplexing interfaces. There are 4 types of interfaces that together constitute the Protection Group, each with its own role:

- Normal, or protected interface(s) – this interface(s) abstract the Line(s), or Multiplexing Section(s) being protected. In a 1+1 scheme, there is one such interface; in a 1:N scheme there are N such interfaces.
- Extra-Traffic interface – this interface abstract the transporting of the extra traffic, which may be carried in a 1:N Trail Protection scheme.
- Working interface – this is the interface on which the regular traffic is transported on. If it fails or underperforms, traffic is switched to the "protecting interface".
- Protecting interface – this is the interface that traffic is switched to in case the "working interface", or layers below fail, or underperform.

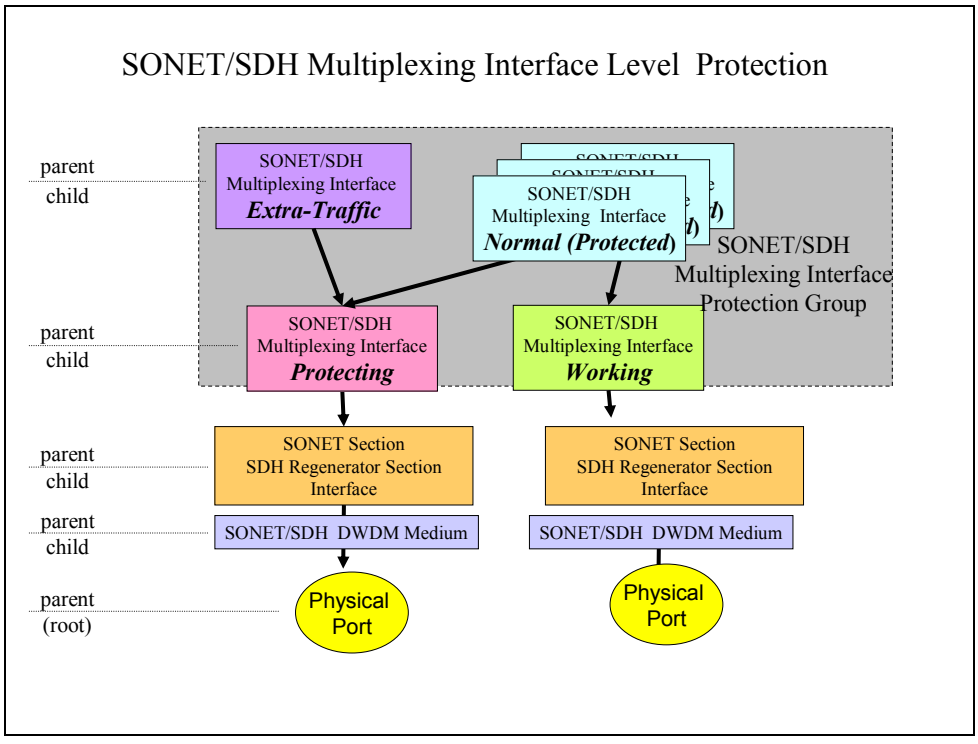

Figure 2-5 SONET Line, or SDH Multiplexing Section Protection

```
/*
** SONET-SDH Interface Protection Information
*/
```

```
typedef struct {
        NPF_IfSONET_SDH_Protection_Flags_t    ProtFlags;
        NPF_IfSONET_SDH_Protection_Type_t     ProtType;
        NPF_IfSONET_SDH_Protection_Arch_t     ArchType;
        NPF_IfSONET_SDH_Protection_Switch_t   SwitchType;
        NPF_IfSONET_SDH_Protection_Operation_t  OperType;
        NPF_boolean_t NPF_IfSONET_SDH_Protection_Signal_t;
        } NPF_IfSONET_SDH_Protection_t;



typedef enum {

      NPF_IF_SONET_SDH_WORKING = 1,   /* Working Trail of a APS combination */
      NPF_IF_SONET_SDH_PROTECTING = 2, /* The Protecting Trail of a APS combination */
      NPF_IF_SONET_SDH_NORMAL = 3, /* This Multiplex Section is protected */
      NPF_IF_SONET_SDH_EXTRA_TRAFFIC=4 /* This Multiplex Section carries extra-traffic */

} NPF_IfSONET_SDH_Protection_Flags_t;

typedef enum {

      NPF_IF_SONET_SDH_TRAIL_PROT = 1,       /* Trail Protection */
      NPF_IF_SONET_SDH_SUBNETWORK = 2 /* Subnet Protection */

} NPF_IfSONET_SDH_Protection_Type_t;

typedef enum {

      NPF_IF_SONET_SDH_ONE_ONE = 1,   /* 1+1 Protection */
      NPF_IF_SONET_SDH_ONE_N = 2      /* 1:N Protection */

} NPF_IfSONET_SDH_Protection_Arch_t;

typedef enum {

      NPF_IF_SONET_SDH_UNI_D = 1,     /* Unidirectional */
      NPF_IF_SONET_SDH_BI_D = 2       /* Bidirectional */

} NPF_IfSONET_SDH_Protection_Switch_t;

typedef enum {

      NPF_IF_SONET_SDH_REVERTIVE = 1,        /*  */
      NPF_IF_SONET_SDH_NON_REVERTIVE = 2     /*  */

} NPF_IfSONET_SDH_Protection_Operation_t;
```

2.1.2.5 SONET/SDH Adaptation Interface Attributes

SONET or SDH aligning, or adaptation is a procedure by which the frame offset information (pointer) is incorporated into the SONET Virtual Tributary (VT), or SONET STS, respectively SDH Tributary Unit (TU) or the SDH Administrative Unit (AU) when adapting to the frame reference of the supporting line layer.

The SONET/SDH Adaptation Interface is abstracting the adapting functions of SONET/SDH between the path layer and line or multiplexing section layer (1), or between the low order path and high order path layer (2). Consequently, there are two SONET/SDH adapting interface types: high order (1) and low order (2) adapting interfaces.

SONET/SDH Adaptation Interfaces are in a parent-child relationship with Multiplexing interfaces (lower layer), and Mapping Interface (higher layer). They are children of Multiplexing interfaces and parents of Mapping interfaces.

```
/*
**      SONET-SDH  Adaption Interface attributes,
**      used in the union within NPF_IfGeneric_t.
*/
typedef struct {


        union {

        NPF_IfSDH_HO_AdaptIf_t SDH_HO_Adaptation;
        NPF_IfSDH_LO_AdaptIf_t SDH_LO_Adaptation;
        NPF_IfSONET_STS_AdaptIf_t     SONET_STS_Adaptation;
        NPF_IfSONET_VT_AdaptIf_t      SONET_VT_Adaptation;


        } u;
        NPF_IfSONET_SDH_Adapt_FMConf_t   Adapt_FM_Conf;
       NPF_IfSONET_SDH_AdaptStatus_t    AdaptationStatus;

                } NPF_IfSONET_SDH_Adaptation_t;
```

The elements of the Fault Management configuration and status structure are defined in the Fault Management Section. Elements of Performance Monitoring are defined in the Performance monitoring section.

2.1.2.5.1 SDH High Order Adaptation Interface Attributes

The High Order Adaptation interface is abstracting the adaptation of SDH Virtual Containers to multiplexing in the corresponding SDH Administrative Unit Groups (see Figure 2-6).

The following Adaptation is supported:

- SDH
  - AU-4-256c – Administrative Unit-4-256c. Adaptation interface is abstracting the adaptation of a VC-4-256c type virtual container to a AU-4-256c.
  - AU-4-64c – Administrative Unit-4-64c. Adaptation interface is abstracting the adaptation of a VC-4-64c type virtual container to a AU-4-64c.
  - AU-4-16c – Administrative Unit-4-16c. Adaptation interface is abstracting the adaptation of a VC-4-16c type virtual container to a AU-4-16c.
  - AU-4-4c – Administrative Unit-4-4c. Adaptation interface is abstracting the adaptation of a VC-4-4c type virtual container to a AU-4-4c.

- AU-4 – Administrative Unit-4 adaptation interface is abstracting the adaptation of a VC-4 type virtual container to a AU-4.
- AU-3 – Administrative Unit-3 adaptation interface is abstracting the adaptation of a VC-3 type virtual container to a AU-3.

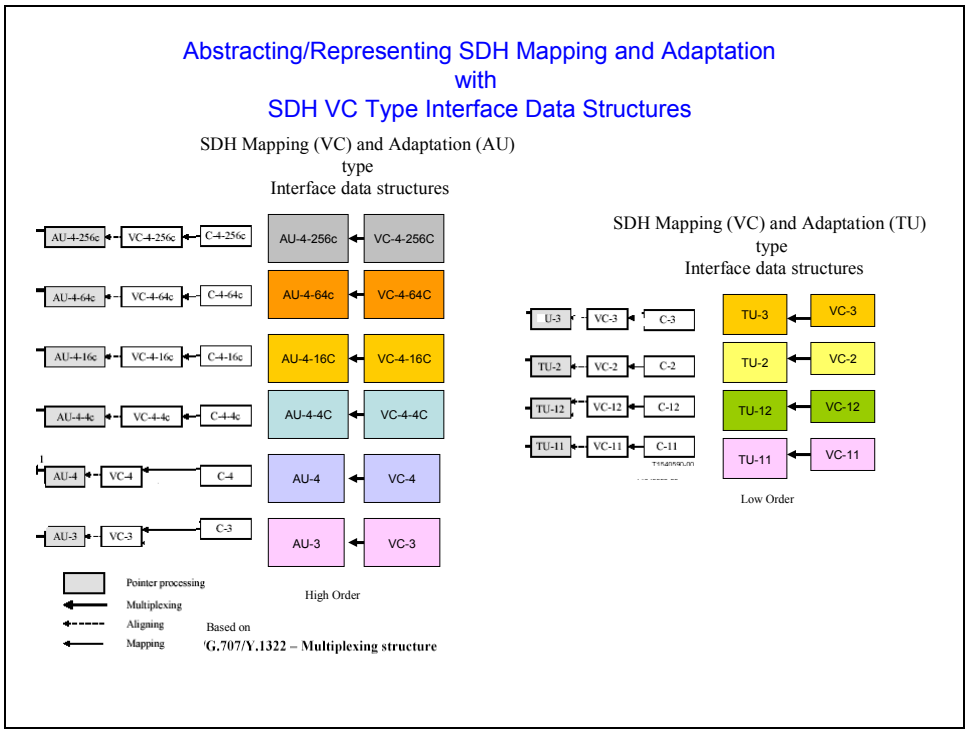

Figure 2-6 SDH Adaptation and Mapping Interfaces

The following data structures are defined:

```
/*
** SONET High Order Adaptation Attributes
*/


typedef enum {

  NPF_IF_SDH_Adapt_none = 0,  /* Type of adaptation not defined */
  NPF_IF_SDH_Adapt_AU_3 = 1,          /* AU-3  51.84Mbps */
  NPF_IF_SDH_Adapt_AU_4 = 2,          /* AU-4  155.55Mbps */
  NPF_IF_SDH_Adapt_AU_4_4c = 3,       /* AU-4-4c 311.04Mbps */
  NPF_IF_SDH_Adapt_AU_4_16c = 4,      /* AU-4-16c 622.08Mbps */
  NPF_IF_SDH_Adapt_AU_4_64c = 5,      /* AU-4-64c 2.48Gbps */
  NPF_IF_SDH_Adapt_AU_4_256c = 6      /* AU-4-256c 39.68Gbps */

} NPF_IfSDH_HO_AdaptIf_t
```

2.1.2.5.2 SONET STS Adaptation Interface Attributes

The SONET STS Adaptation interface is abstracting the adaptation of the high order SONET STS Synchronous Payload Envelopes to multiplexing in the SONET Synchronous TS Groups (see Figure 2-2).

- SONET
  - STS-768c – A STS-768c adaptation interface is abstracting the adaptation of a STS-768c SPE to multiplexing in a STS-768 Group.
  - STS-192c – A STS-192c adaptation interface is abstracting the adaptation of a STS-192c SPE to multiplexing in a STS-192 Group.
  - STS-48c – A STS-48c adaptation interface is abstracting the adaptation of a STS-48c SPE to multiplexing in a STS-48 Group.
  - STS-12c – A STS-12c adaptation interface is abstracting the adaptation of a STS-12c SPE to multiplexing in a STS-12 Group.
  - STS-3c – A STS-3c adaptation interface is abstracting the adaptation of a STS-3c SPE to multiplexing in a STS-3 Group.
  - STS-1 – A STS-1 adaptation interface is abstracting the adaptation of a STS-1 SPE to multiplexing in a STS-1 Group

The following data structures are defined:

```
/*
** SONET STS (High Order) Adaptation Attributes
*/


typedef enum {

  NPF_IF_SONET_Adapt_none = 0,        /* Type of adaptation not defined */

  NPF_IF_SONET_Adapt_STS_1 = 1
                                      /* Adapted SPE is STS1 51.84Mbps */
  NPF_IF_SONET_Adapt_STS_3c = 2,
                                      /* Adapted SPE is STS3c 155.55Mbps */

  NPF_IF_SONET_Adapt_STS_12c = 3,
                                      /* Adapted SPE is STS12c 622.08Mbps */
  NPF_IF_SONET_Adapt_STS_48c = 4,
                                      /* Adapted SPE is STS48c 2.48Gbps */
  NPF_IF_SONET_Adapt_STS_192c = 5,
                                      /* Adapted SPE is STS192c 9.92Gbps */
  NPF_IF_SONET_Adapt_STS_768c = 6
                                      /* Adapted SPE is STS768c 39.68Gbps */

} NPF_IfSONET_STS_AdaptIf_t;
```

2.1.2.5.3 SDH Low Order Adaptation Interface Attributes

The SDH Low Order Adaptation interface is abstracting the adaptation functions of SDH low order path Virtual Containers to high order path. Functionally this consist in

generating and adding frame offset information (pointer) to the VC. The Tributary Unit structures contain information for this adaptation.

The following adaptation is supported (see Figure 2-6):

- SDH
  - Tributary Unit 3: TU-3
  - Tributary Unit 2: TU-2
  - Tributary Unit 12: TU-12
  - Tributary Unit 11: TU-11

```
In terms of framing of information, TUs contain pointers and VCs.

/*
** SDH Low Order Adaptation Interface Attributes
*/
typedef enum {

  NPF_IF_SDH_Adapt_TU-11 = 7, /* Adapt Low Order VC-11 - TU-11 */
  NPF_IF_SDH_Adapt_TU-12 = 8, /* Adapt Low Order VC-12 - TU-12 */
  NPF_IF_SDH_Adapt_TU-2 = 9,  /* Adapt Low Order VC-2 - TU-2 */
  NPF_IF_SDH_Adapt_TU-3 = 10  /* Adapt Low Order VC-3 - TU-3 */

      } NPF_IfSDH_LO_AdaptIf_t;
```

2.1.2.5.4 SONET VT Adaptation Interface

The SONET VT Adaptation interface is abstracting the adaptation functions of low order SONET Virtual Tributaries Synchronous Payload Envelopes (VT SPEs) into SONET Virtual Tributaries. Functionally it consists of adding frame offset information (pointer) to the VT SPE (see Figure 2-4).

- SONET
  - Virtual Tributary 6 - VT6
  - Virtual Tributary 3 - VT3
  - Virtual Tributary 2 – VT2
  - Virtual Tributary 1.5 – VT1.5

The following data structures are defined:

```
/*
** SONET VT Adaptation Interface
*/

typedef enum {

  NPF_IF_SONET_VT15 = 7,      /* SONET VT1.5 */
  NPF_IF_SONET_VT2 = 8,       /* SONET VT2 */
  NPF_IF_SONET_VT3 = 9,       /* SONET VC3 */
  NPF_IF_SONET_VT6 = 10       /* SONET VT6 */

      } NPF_IfSONET_VT_AdaptIf_t;
```

2.1.2.6 SONET/SDH Mapping Interface Attributes

SONET or SDH mapping is a procedure by which tributaries are adapted into SONET Virtual Tributary SPEs or SONET STS SPEs, respectively SDH Virtual Containers, at the boundary of a SONET, or SDH network (path layer termination).

The SONET/SDH Mapping Interface is abstracting the mapping functions of SONET/SDH. There are two SONET/SDH mapping interface types: high order and low order mapping interfaces:

High Order SONET/SDH Mapping interfaces have a parent-child relationship with Adaptation interfaces (layer below). In this case they are children of Adaptation interfaces. They may have a parent-child relationship with Low Order Multiplexing Interfaces (layer above). In this case they are parents of the Low Order Multiplexing interfaces (see Figure 2-2).

Low Order Mapping interfaces have a parent-child relationship with Adaptation Interfaces (layer below). They are children of the Adaptation interface.

The type of user payload mapped by a Mapping interface is indicated by the type of interface that is in a parent-child relationship with the Mapping interface. For instance an ATM cell mapping on SONET/SDH, will have an ATM interface in a parent-child relationship with the SONET/SDH Mapping interface (the ATM interface being the child of the SONET/SDH interface).

```
/*
**      SONET-SDH  Mapping Interface attributes,
**
*/
typedef struct {


        union {
        NPF_IfSDH_HO_MapIf_t          SDH_HO_Mapping;       /* SDH High Order */
        NPF_IfSDH_LO_MapIf_t          SDH_LO_Mapping;       /* SDH Low Order */
        NPF_IfSONET_STS_MapIf_t       SONET_STS_Mapping;    /* SONET STS */
        NPF_IfSONET_VT_MapIf_t        SONET_VT_Mapping;     /* SONET VT */

        } u;

        NPF_IfSONET_SDH_Protection_t  Map_Protection;
        NPF_IfSONET_SDH_VCG_Attrib_t  VCGAttrib;
        NPF_IfSONET_SDH_MapStatus_t   MapStatus;
                                                  /* Path Current Status */

                } NPF_IfSONET_SDH_Mapping_t;
```

Note: SDH High Order Path is corresponding to SONET STS Level Path.

The elements of the fault management configuration and status structures are defined in the Fault Management Section.

2.1.2.6.1 SDH High Order Mapping Interface Attributes

The High Order Mapping interface is abstracting the mapping of payloads (SDH Containers) to high order SDH Virtual Containers (VCs).

The following mapping is supported (see Figure 2-3):

- SDH
  - VC-4-256c – Virtual Container VC-4-256c mapping interface is abstracting the mapping of a C-4-256c container to a VC-4-256c type virtual container.
  - VC-4-64c – Virtual Container VC-4-64c mapping interface is abstracting the mapping of a C-4-64c container to a VC-4-64c type virtual container.
  - VC-4-16c – Virtual Container VC-4-16c mapping interface is abstracting the mapping of a C-4-16c container to a VC-4-16c type virtual container.
  - VC-4-4c – Virtual Container VC-4-4c mapping interface is abstracting the mapping of a C-4-4c container to a VC-4-4c type virtual container.
  - VC-4 – Virtual Container VC-4 mapping interface is abstracting the mapping of a C-4 container to a VC-4 type virtual container.
  - VC-3 – Virtual Container VC-3 mapping interface is abstracting the mapping of a C-3 container to a VC-3 type virtual container.

The SDH High Order Mapping Interface abstracts also Virtual Concatenation. A Virtual Concatenation Group (VCG) is a Mapping Interface, which has a number of Parent Mapping Interfaces. The bandwidth provided by the VCG Mapping interface is the sum of the bandwidth of all the parent Mapping interfaces.

The following data structure is defined:

```
/*
** SDH High Order Mapping Attributes
*/


typedef enum {

  NPF_IF_SDH_Map_VC_3 = 1,           /* VC-3  51.84Mbps */
  NPF_IF_SDH_Map_VC_4 = 2,           /* VC-4  155.55Mbps */
  NPF_IF_SDH_Map_VC_4-4c = 3,        /* VC-4-4c 311.04Mbps */
  NPF_IF_SDH_Map_VC_4-16c = 4,       /* VC-4-16c 622.08Mbps */
  NPF_IF_SDH_Map_VC_4-64c = 5,       /* VC-4-64c 2.48Gbps */
  NPF_IF_SDH_Map_VC_4-256c = 6,      /* VC-4-256c 39.68Gbps */

  NPF_IF_SDH_Map_VCG = 255           /* VCG, bandwidth provided by the summ of the */
```

```
                                        /* bandwidth of all parent Mapping interfaces */
} NPF_IfSDH_HO_MapIf_t
```

2.1.2.6.2 SONET STS Mapping Interface

The SONET STS Mapping interface is abstracting the mapping of user payloads to SONET STS Synchronous Payload Envelopes (STS SPEs) .

The following mapping is supported (see Figure 1-6):

- SONET
  - STS-768c – A STS Synchronous Payload Envelope 768c mapping interface is abstracting the mapping of a user payload to a STS-256c SPE.
  - STS-192c – A STS Synchronous Payload Envelope 192c mapping interface is abstracting the mapping of a user payload to a STS-192c SPE.
  - STS-48c – A STS Synchronous Payload Envelope 48c mapping interface is abstracting the mapping of a user payload to a STS-48c SPE.
  - STS-12c – A STS Synchronous Payload Envelope 12c mapping interface is abstracting the mapping of a user payload to a STS-12c SPE.
  - STS-3c – A STS Synchronous Payload Envelope 3c mapping interface is abstracting the mapping of a user payload to a STS-3c SPE
  - STS-1 – A STS Synchronous Payload Envelope 1 mapping interface is abstracting the mapping of a user payload to a STS-1 SPE

The SONET STS Mapping Interface abstracts also Virtual Concatenation. A Virtual Concatenation Group (VCG) is a Mapping Interface, which has a number of Parent Mapping Interfaces. The bandwidth provided by the VCG Mapping interface is the sum of the bandwidth of all parent Mapping interfaces.

The following data structures are defined:

```
/*
** SONET High Order Mapping Attributes
*/


typedef enum {

  NPF_IF_SONET_Map_STS_1_SPE = 1,     /* SPE is STS1 51.84Mbps */
  NPF_IF_SONET_Map_STS_3c_SPE = 2,    /* SPE is STS3c 155.55Mbps */
  NPF_IF_SONET_Map_STS_12c_SPE = 3,   /* SPE is STS12c 622.08Mbps */
  NPF_IF_SONET_Map_STS_48c_SPE = 4,   /* SPE is STS48c 2.48Gbps */
  NPF_IF_SONET_Map_STS_192c_SPE = 5,  /* SPE is STS192c 9.92Gbps */
  NPF_IF_SONET_Map_STS_768c_SPE = 6,  /* SPE is STS768c 39.68Gbps */

  NPF_IF_SONET_Map_VCG = 255          /* VCG, bandwidth provided by the summ of the */
                                      /* bandwidth of all parent Mapping interfaces */
} NPF_IfSONET_STS_MapIf_t;
```

2.1.2.6.3 SDH Low Order Mapping Interface

The Low Order Mapping interface is abstracting the mapping functions of payloads (SDH Containers) into low order SDH Virtual Containers (VCs).

The following mapping is supported (see Figure 2-6):

- SDH
  - Virtual Container 2: VC-2
  - Virtual Container 12: VC-12
  - Virtual Container 11: VC-11

The SDH Low Order Mapping Interface abstracts also Virtual Concatenation. A Virtual Concatenation Group (VCG) is a Mapping Interface, which has a number of Parent Mapping Interfaces. The bandwidth provided by the VCG Mapping interface is the sum of the bandwidth of all the parent Mapping interfaces

The following data structure is defined:

```
/*
** SDH Low Order Mapping Interface Attributes
*/
typedef enum {


  NPF_IF_SDH_Map_VC11 = 7,     /* Map Low Order VC11 */
  NPF_IF_SDH_Map_VC12 = 8,     /* Map Low Order VC12 */
  NPF_IF_SDH_Map_VC2 = 9,      /* Map Low Order VC2 */

  NPF_IF_SDH_Map_LO_VCG = 255         /* VCG, bandwidth provided by the summ of the */
                              /* bandwidth of all parent Mapping interfaces */
} NPF_IfSDH_LO_MapIf_t;
```

2.1.2.6.4 SONET VT Mapping Interface Attributes

The VT Mapping interface is abstracting the mapping functions of user payloads to SONET Virtual Tributaries Synchronous Payload Envelopes (VT SPEs).

The following mapping is supported (see Figure 2-4):

- SONET
  - Virtual Tributary 6 SPE - VT6 SPE
  - Virtual Tributary 3 SPE - VT3 SPE
  - Virtual Tributary 2 SPE – VT2 SPE
  - Virtual Tributary 1.5 SPE – VT1.5 SPE

The SONET VT Mapping Interface abstracts also Virtual Concatenation. A Virtual

Concatenation Group (VCG) is a Mapping Interface, which has a number of Parent Mapping Interfaces. The bandwidth provided by the VCG Mapping interface is the sum of the bandwidth of all parent Mapping interfaces.

The following data structures are defined:

```
/*
** SONET VT Mapping Interface
*/

typedef enum {

  NPF_IF_SONET_VT15_SPE = 7,          /* SONET VC11 */
  NPF_IF_SONET_VT2_SPE = 8,           /* SONET VC12 */
  NPF_IF_SONET_VT3_SPE = 9,           /* SONET VC3 */
  NPF_IF_SONET_VT_6_SPE = 10,         /* SONET VT6 */

   NPF_IF_SONET_Map_LO_VCG = 255      /* VCG, bandwidth provided by the sum of the */
                              /* bandwidth of all parent Mapping interfaces */
} NPF_IfSONET_VT_MapIf_t;
```

2.1.2.6.5 SONET/SDH Virtual Concatenation Group Attributes

The SONET/SDH Virtual Concatenation Group (VCG) attributes are:

- Trail Signal Degrade (TSD) Enable/Disable
- LCAS Enable/Disable

The data structure is:

```
/*
** VCG Attributes
*/
typedef struct {

 NPF_boolean_t  NPF_IF_SONET_SDH_TSD_ENABLE   /* TSD Enable */
 NPF_boolean_t  NPF_IF_SONET_SDH_LCAS_ENABLE  /* LCAS Enable*/

} NPF_IfSONET_SDH_VCG_Attrib_t;
```

2.1.2.6.6 SONET/SDH Mapping Interface Level Protection

The SONET/SDH Path Trail Protection is abstracted by one additional level of interfaces (Figure 2-7.)

The interfaces on this level are still Mapping interfaces. There are 4 types of interfaces that together constitute the Protection Group, each with its own role:

- Normal, or protected interface(s) – this interface(s) abstract the Line(s), or Multiplexing Section(s) being protected. In a 1+1 scheme, there is one such interface; in a 1:N scheme there are N such interfaces.
- Extra-Traffic interface – this interface abstract the transporting of the extra traffic, which may be carried in a 1:N Trail Protection scheme.

- Working interface – this is the interface on which the regular traffic is transported on. If it fails or underperforms, traffic is switched to the "protecting interface".
- Protecting interface – this is the interface that traffic is switched to in case the "working interface", or layers below fail, or underperform.

The data structure defining protection is identical with that defined for Multiplexing Interfaces.

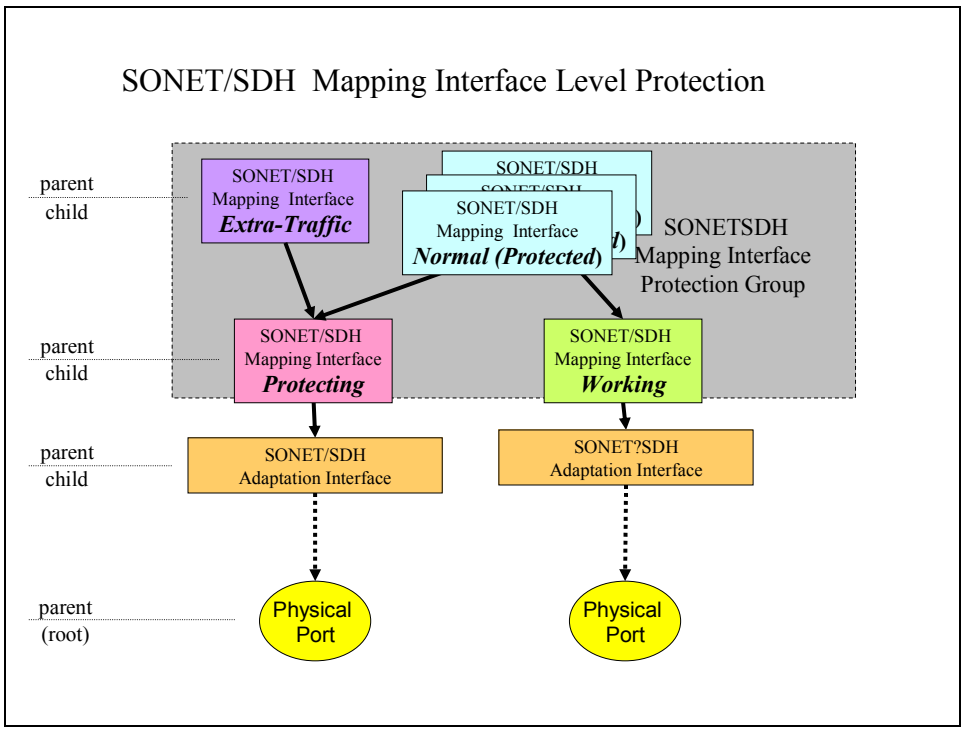

Figure 2-7 SONET/SDH Mapping Interface Level Protection

2.1.3 SONET SONET/SDH Fault Management Data Structures

Fault Management Information is propagated as follows:

- Between SONET/SDH distinct layers of the SONET/SDH stack on the same node (network element). Alarm Indication Signal is a typical Fault Management information, which is passed on from layer to layer (See Figure 2-8).

For example:

- AIS-L – Line Alarm Indication Signal

- AIS-P – STS Path Alarm Indication Signal
- AIS-V – VT Path Alarm Indication Signal
- DS0 Path AIS – DS0 Path Alarm Indication Signal

- Between SONET/SDH same layers on distinct nodes (network elements)

For example:

- RDI-L – Line Remote Defect Information
- RDI-P – STS Path Remote Defect Information
- RDI-V – VT Path Remote Defect Information
- RFI-V - VT Path Remote Failure Indication

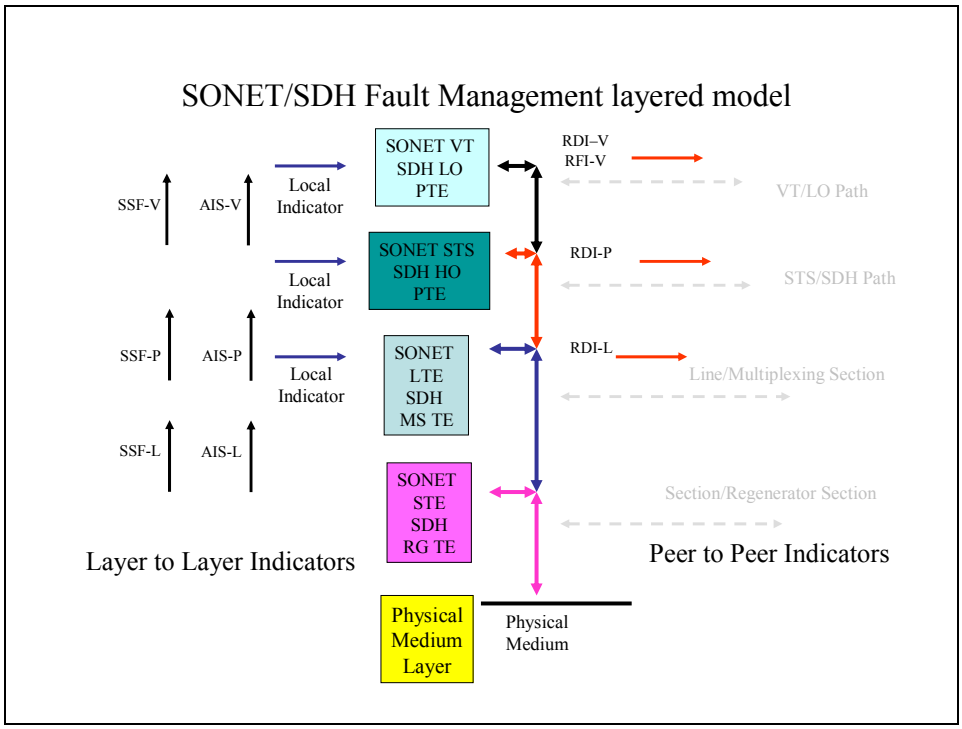

Figure 2-8 Fault Management Information Propagation

2.1.3.1 SONET/SDH Fault Management Configuration Structures

The following structures were defined:

2.1.3.1.1 SONET/SDH Medium Interface FM Configuration Attributes

```
/*
** SONET-SDH  Medium Interface Fault Management Configuration Parameters
*/

typdef struct{
```

```
        NPF_IfSONET_SDH_TP_Mode_t              TPMode; /* Trail Termination monitoring */
        NPF_IfSONET_SDH_SSF_ReportMode_t SSF_ReportMode; /* Server Signal Fail */
        NPF_IfSONET_SDH_AIS_ReportMode_t AIS_ReportMode; /* Alarm Indication Signal */
        NPF_IfSONET_SDH_RDI_ReportMode_t RDI_ReportMode; /* Remote Defect Information */
        } NPF_IfSONET_SDH_Medium_FMConfig_t;

/*
** Configuration for SONET-SDH  Trail Termination Functions
*/


typedef enum {
        NPF_IF_SONET_SDH_NonMonitoringState = 0,
                                        /* Interface in non monitoring mode
                                        ** for Trail Termination functions
                                        ** (Tpmode = NMON)
                                        */
        NPF_IF_SONET_SDH_MonitoringState = 1
                                        /* Interface in monitoring mode
                                        ** for Trail Termination functions
                                        ** (Tpmode = MON)
                                        */
                                } NPF_IfSONET_SDH_TP_Mode_t;

/*
** Configuration SONET-SDH  Server Signal Failure reporting
*/

typedef enum {
        NPF_IF_SONET_SDH_SSF_ReportDisable = 0,
                                        /* Interface configured not to report
                                        ** Server Signal Failure
                                        */
        NPF_IF_SONET_SDH_SSF_ReporteEnable = 1
                                        /* Interface configured to report SSF */
                } NPF_IfSONET_SDH_SSF_ReportMode_t;
/*
**  Configuration of SONET-SDH  Alarm Indication Signal (AIS) reporting Mode
*/

typedef enum {
        NPF_IF_SONET_SDH_AIS_ReportDisable = 0,
                                        /* Interface configured not to report AIS signal */
        NPF_IF_SONET_SDH_AIS_ReportEnable = 1
                                        /* Interface configured to report AIS signal */
                } NPF_IfSONET_SDH_AIS_ReportMode_t;

/*
** Configuration SONET-SDH  Remote Defect Information reporting
*/
typedef enum {
        NPF_IF_SONET_SDH_RDI_ReportDisable = 0,
                /* Inteface configured not to report Remote Defect Information */
        NPF_IF_SONET_SDH_RDI_ReporteEnable = 1
                /* Interface configured to report Remote Defect Information */
                } NPF_IfSONET_SDH_RDI_ReportMode_t;
```

2.1.3.1.2 SONET Section, SDH Reg. Section Interface FM Configuration

```
typedef struct {


      NPF_IfSONET_SDH_Section_AIS_RepMode_t          AIS_SecReportMode;
                                              /* Alarm Indication Signal */
      NPF_IfSONET_SDH_Section_RDI_RepMode_t          RDI_SecReportMode;
                                              /* Remote Defect Information */
```

```
      NPF_IfSONET_SDH_Section_SSF_RepMode_t          SSF_SecReportMode;
                                              /* Server Signal Fail */
      NPF_IfSONET_SDH_Section_TIMCfgSet_t            Section_TimCfgSet;
                              /* Section/RS TIM processing enabled/disabled */
      NPF_IfSONET_SDH_Section_TTI_AISCfgSet_t        Section_TTIAisCfgSet;
                      /* AIS Insertion on Section/RS TTI and AIS enabled/disabled */
      /*
      ** FEC and Delay buffers are to be used only from OC-48 and up.
      */
      NPF_IfSONET_SDH_Section_Tx_FECCfgSet_t         Section_Tx_FECCfgSet_t;
      NPF_IfSONET_SDH_Section_Rx_FECCfgSet_t         Section_Rx_FECCfgSet_t;
      NPF_IfSONET_SDH_Section_Tx_DelayCfgSet_t       Section_Tx_DelCfgSet_t;
      NPF_IfSONET_SDH_Section_Rx_DelayCfgSet_t       Section_Rx_DelCfgSet_t;

      }NPF_IfSONET_SDH_Section_FMConf_t;


/*
** Configuration SONET-SDH  Server Signal Failure reporting
*/

typedef enum {
       NPF_IF_SONET_SDH_SECTION_SSF_ReportDisable = 0,
                                     /* Interface configured not to report
                                     ** Server Signal Failure
                                     */
       NPF_IF_SONET_SDH_SECTION_SSF_ReporteEnable = 1
                                     /* Interface configured to report SSF */
              } NPF_IfSONET_SDH_Section_SSF_RepMode_t;
/*
**  Configuration of SONET-SDH  Alarm Indication Signal (AIS) reporting Mode
*/

typedef enum {
       NPF_IF_SONET_SDH_SECTION_AIS_ReportDisable = 0,
                                     /* Interface configured not to report AIS signal */
       NPF_IF_SONET_SDH_SECTION_AIS_ReportEnable = 1
                                     /* Interface configured to report AIS signal */
              } NPF_IfSONET_SDH_Section_AIS_RepMode_t;

/*
** Configuration SONET-SDH Remote Defect Information reporting
*/
typedef enum {
       NPF_IF_SONET_SDH_SECTION_RDI_ReportDisable = 0,
              /* Inteface configured not to report Remote Defect Information */
       NPF_IF_SONET_SDH_SECTION_RDI_ReporteEnable = 1
              /* Interface configured to report Remote Defect Information */
              } NPF_IfSONET_SDH_Section_RDI_RepMode_t;

typedef enum {
      NPF_IF_SONET_SDH_SECTOIN_TIMdis = 0, /* RS / Section TIM processing disabled */
      NPF_IF_SONET_SDH_SECTION_TIMen = 1 /* RS / Section TIM processing enabled */
      } NPF_IfSONET_SDH_Section_TIMCfgSet_t ;

typedef enum {
      NPF_IF_SONET_SDH_SECTION_TTIAISdis = 0,
              /* AIS insertion on Regenerator Section TTI mismtach disabled */
      NPF_IF_SONET_SDH_SECTION_TTIAISen = 1
              /* AIS insertion on Regenerator Section TTI mismtach disabled */
       } NPF_IfSONET_SDH_Section_TTI_AisCfgSet_t ;

typedef enum {
      NPF_IF_SONET_SDH_SECTION_TxFECOff = 0, /* FEC Calculation on Tx not enabled */
      NPF_IF_SONET_SDH_SECTION_TxFECOn = 1   /* FEC Calculation on Tx Enabled */
      } NPF_IfSONET_SDH_Section_Tx_FECCfgSet_t

typedef enum {
      NPF_IF_SONET_SDH_SECTION_RxFECOff = 0, /* FEC Calculation on Rx not enabled */
      NPF_IF_SONET_SDH_SECTION_RxFECOn = 1   /* FEC Calculation on Rx Enabled */
```

```
      } NPF_IfSONET_SDH_Section_Rx_FECCfgSet_t

typedef enum {
      NPF_IF_SONET_SDH_Section_TxDelayOff = 0,       /* Tx Delay Buffers not used */
      NPF_IF_SONET_SDH_Section_TxDelay = 1           /* Tx Delay Buffers used */
      } NPF_IfSONET_SDH_Section_Tx_DelayCfgSet_t


typedef enum {
      NPF_IF_SONET_SDH_Section_RxDelayOff = 0,       /* Rx Delay Buffers not used */
      NPF_IF_SONET_SDH_Section_RxDelay = 1           /* Rx Delay Buffers used */
      } NPF_IfSONET_SDH_Section_Rx_DelayCfgSet_t
```

2.1.3.1.3 SONET/SDH Multiplexing Interface Fault Management Configuration

```
Multiplexing interfaces are associated with Line Termination.


/*
** AIS-L Line Alarm Indication Signal – Indicate to Line Terminating Equipment (LTE) that
** Section Terminating Equipment (STE) detected a signal defect.
**
** AIS-L generated on detection by STE of LOS (loss of signal) or LOF (loss of frame).
**
** RDI-L Line Remote Defect Indication – Indicate to transmitting Line
** Terminating Equipment (LTE) that the receiving LTE detected an incoming line defect.
**
** RDI-L generated on detection by LTE of LOS (loss of signal) or LOF (loss of frame), or
** AIS-L.
**/


typedef struct {

      NPF_IfSONET_SDH_Mux_AIS_RepMode_t              AIS_MuxReportMode;
                                          /* Alarm Indication Signal */
      NPF_IfSONET_SDH_Mux_RDI_RepMode_t              RDI_MuxReportMode;
                                          /* Remote Defect Information */
      NPF_IfSONET_SDH_Mux_SSF_RepMode_t              SSF_MuxReportMode;
                                          /* Server Signal Fail */

      NPF_IfSONET_SDH_MuxThresh_t            /* Threshold */

      }NPF_IfSONET_SDH_Mux_FMConf_t;


/*
** Configuration SONET-SDH  Server Signal Failure reporting
*/

typedef enum {
       NPF_IF_SONET_SDH_MUX_SSF_ReportDisable = 0,
                                    /* Interface configured not to report
                                    ** Server Signal Failure
                                    */
       NPF_IF_SONET_SDH_SSF_ReporteEnable = 1
                                    /* Interface configured to report SSF */
              } NPF_IfSONET_SDH_Mux_SSF_RepMode_t;
/*
**  Configuration of SONET-SDH  Alarm Indication Signal (AIS) reporting Mode
*/

typedef enum {
       NPF_IF_SONET_SDH_MUX_AIS_ReportDisable = 0,
                                    /* Interface configured not to report AIS signal */
       NPF_IF_SONET_SDH_MUX_AIS_ReportEnable = 1
                                    /* Interface configured to report AIS signal */
```

```
              } NPF_IfSONET_Mux_SDH_AIS_RepMode_t;

/*
** Configuration SONET-SDH  Remote Defect Information reporting
*/
typedef enum {
       NPF_IF_SONET_SDH_MUX_RDI_ReportDisable = 0,
              /* Inteface configured not to report Remote Defect Information */
       NPF_IF_SONET_SDH_MUX_RDI_ReporteEnable = 1
              /* Interface configured to report Remote Defect Information */
              } NPF_IfSONET_SDH_Mux_RDI_RepMode_t;

/*
**  SONET-SDH  Mapping Interface Signal Degeneration Threshold
*/
typedef enum {

  NPF_IF_SONET_SDH_ThreshBursty = 5,

  NPF_IF_SONET_SDH_ThreshPoisson = 6

       } NPF_IfSONET_SDH_MuxThresh_t;
```

2.1.3.1.4 SONET/SDH Adaptation Interface Fault Management Configuration

```
typedef struct {

      NPF_IfSONET_SDH_Adapt_AIS_RepMode_t                  AIS_AdaptReportMode;
                                          /* Alarm Indication Signal */
      NPF_IfSONET_SDH_Adapt_RDI_RepMode_t          RDI_AdaptReportMode;
                                          /* Remote Defect Information */
      NPF_IfSONET_SDH_Adapt_SSF_RepMode_t          SSF_AdaptReportMode;
                                          /* Server Signal Fail */

      }NPF_IfSONET_SDH_Adapt_FMConf_t;



/*
** Configuration SONET-SDH  Server Signal Failure reporting
*/

typedef enum {
       NPF_IF_SONET_SDH_ADAPT_SSF_ReportDisable = 0,
                                    /* Interface configured not to report
                                    ** Server Signal Failure
                                    */
       NPF_IF_SONET_SDH_ADAPT_SF_ReporteEnable = 1
                                    /* Interface configured to report SSF */
              } NPF_IfSONET_SDH_Adapt_SSF_RepMode_t;
/*
**  Configuration of SONET-SDH  Alarm Indication Signal (AIS) reporting Mode
*/

typedef enum {
       NPF_IF_SONET_SDH_ADAPT_AIS_ReportDisable = 0,
                                    /* Interface configured not to report AIS signal */
       NPF_IF_SONET_SDH_ADAPT_IS_ReportEnable = 1
                                    /* Interface configured to report AIS signal */
              } NPF_IfSONET_SDH_Adapt_AIS_RepMode_t;

/*
** Configuration SONET-SDH  Remote Defect Information reporting
*/
typedef enum {
       NPF_IF_SONET_SDH_ADAPT_DI_ReportDisable = 0,
              /* Inteface configured not to report Remote Defect Information */
       NPF_IF_SONET_SDH_ADAPT_RDI_ReporteEnable = 1
```

```
              /* Interface configured to report Remote Defect Information */
              } NPF_IfSONET_SDH_Adapt_RDI_RepMode_t;
```

2.1.3.1.5 SONET/SDH Mapping Interface Fault Management Configuration

The Mapping Interfaces are associated with Path termination.

```
/*
** AIS-P Path Alarm Indication Signal – Indicate to Path Terminating Equipment (PTE)
** that Line Terminating Equipment (LTE) detected one of the following defects:
** LOS, LOF, AIS-L, LOP-P.
**
** RDI-P Path Remote Defect Indication – Indicate to transmitting STS Path
** Terminating Equipment (PTE) that the receiving STS PTE detected an STS defect,
** which is one of the following:
**   - payload defects
**   - server defects
**   - connectivity defects
**
*/


typedef struct {

      NPF_IfSONET_SDH_Map_AIS_MapMode        AIS_MapReportMode;
                                             /* Alarm Indication Signal */
      NPF_IfSONET_SDH_Map_RDI_MapMode               RDI_MapReportMode;
                                             /* Remote Defect Information */
      NPF_IfSONET_SDH_Map_SSF_MapMode               SSF_MapReportMode;
                                             /* Server Signal Fail */

      NPF_IfSONET_SDH_Map_TIMCfgSet_t        Map_timCfgSet;

      NPF_IfSONET_SDH_Map_TTI_AISCfgSet_t    Map_TIMAisCfgSet;

      NPF_IF_SONET_SDH_Map_Thresh_t          Map_ErrThreshold;

       } NPF_IfSONET_SDH_MediumSESthresholdSet_t;



      }NPF_IfSONET_SDH_Map_FMConf_t;

/*
** Configuration SONET-SDH  Server Signal Failure reporting
*/

typedef enum {
       NPF_IF_SONET_SDH_MAP_SSF_ReportDisable = 0,
                                     /* Interface configured not to report
                                     ** Server Signal Failure
                                     */
       NPF_IF_SONET_SDH_MAP_SSF_ReporteEnable = 1
                                     /* Interface configured to report SSF */
              } NPF_IfSONET_SDH_Map_SSF_RepMode_t;
/*
**  Configuration of SONET-SDH  Alarm Indication Signal (AIS) reporting Mode
*/

typedef enum {
       NPF_IF_SONET_SDH_MAP_AIS_ReportDisable = 0,
                                     /* Interface configured not to report AIS signal */
       NPF_IF_SONET_SDH_MAP_AIS_ReportEnable = 1
                                     /* Interface configured to report AIS signal */
              } NPF_IfSONET_SDH_Map_AIS_RepMode_t;
```

```
/*
** Configuration SONET-SDH  Remote Defect Information reporting
*/
typedef enum {
        NPF_IF_SONET_SDH_MAP_RDI_ReportDisable = 0,
                /* Inteface configured not to report Remote Defect Information */
        NPF_IF_SONET_SDH_MAP_RDI_ReporteEnable = 1
                /* Interface configured to report Remote Defect Information */
                } NPF_IfSONET_SDH_Map_RDI_RepMode_t;

typedef enum {
      NPF_IF_SONET_SDH_MAP_TIMdis = 0,  /* TIM processing disabled */
      NPF_IF_SONET_SDH_MAP_TIMen = 1    /*  TIM processing enabled */
      } NPF_IfSONET_SDH_Map_TIMCfgSet_t ;

typedef enum {
      NPF_IF_SONET_SDH_MAP_TTIAISdis = 0,
                      /* AIS insertion on HO Path, TTI mismatch disabled */
      NPF_IF_SONET_SDH_MAP_TTIAISen = 1
                      /* AIS insertion on Ho Path TTI mismatch enabled */
        } NPF_IfSONET_SDH_Map_TTI_AISCfgSet_t ;

/*
**  SONET-SDH  Mapping Interface Signal Degeneration Threshold
*/
typedef enum {

  NPF_IF_SONET_SDH_ThreshBursty = 5,

  NPF_IF_SONET_SDH_ThreshPoisson = 6

        } NPF_IfSONET_SDH_MapThresh_t;
```

2.1.3.2 SONET/SDH Interface Fault Management Status

2.1.3.2.1 SONET/SDH Medium Interface Status

```
/*
** SONET-SDH  Medium Status
*/
typedef enum {

        NPF_IF_SONET_SDH_Medium_NoDefect = 1, /* No Defect */
        NPF_IF_SONET_SDH_Medium_LOS = 2,      /* Loss of Signal Defect */
         NPF_IF_SONET_SDH_Medium_LOF = 3      /* Loss of Frame Defect */

       } NPF_IfSONET_SDH_MediumStatusType_t;

struct NPF_IfSONET_SDH_MediumStatus {
        NPF_uint16_t                          n_data;
        NPF_IfSONET_SDH_MediumStatusType_t   *statusinfo;
        };


/*
** SONET-SDH Medium Current Status
*/

struct NPF_IfSONET_SDH_MediumCurrStatus {
        NPF_uint32_t    MediumTimeElapsed;          /* Time Elapsed */
        NPF_uint32_t    MediumValidIntervals;       /* Valid Intervals */
        NPF_uint32_t    MediumInvalidIntervals;     /* Invalid Intervals */
```

```
};
```

2.1.3.2.2 SONET Section or SDH Regenerator Section Interface Status

```
/*
** SONET-SDH  Section Interface Status
*/


typedef enum {

      NPF_IF_SONET_SDH_Section_NoDefect = 1, /* Section No Defect */
        NPF_IF_SONET_SDH_Section_LOS = 2,     /* Loss of Signal Defect */
         NPF_IF_SONET_SDH_Section_LOF = 3     /* Loss of Frame Defect */

      NPF_IF_SONET_SDH_Section_TIM = 4,       /* Trace Identifier Mismatch Alarm  */
      NPF_IF_SONET_SDH_Section_SSF    = 5     /* Server Signal Failure */

      } NPF_IfSONET_SDH_SectionStatusType_t;

struct NPF_IfSONET_SDH_SectionStatus{
        NPF_uint16_t                          n_data;
        NPF_IfSONET_SDH_SectionStatusType_t   *statusinfo;
        };
```

2.1.3.2.3 SONET/SDH Multiplexing Interface Status

```
/*
**  SONET-SDH  Multiplexing Interface Status
*/


typedef enum {

        NPF_IF_SONET_SDH_Mux_NoDefect = 1,    /* No Defect */
        NPF_IF_SONET_SDH_Mux_AIS = 2,         /* Mux (Line) AIS */
        NPF_IF_SONET_SDH_Mux_RDI = 3,         /* Mux (Line) RDI */
        NPF_IF_SONET_SDH_Mux_SSF = 4,         /* Multiplexing Server Signal Fail */
        /*
        */
        NPF_IF_SONET_SDH_Mux_EXC = 5,         /* Mux Excessive Bit Error Rate */
        NPF_IF_SONET_SDH_Mux_DEG = 6          /* Multiplexing Signal Degrade */

      } NPF_IfSONET_SDH_MuxStatusType_t;

struct NPF_IfSONET_SDH_MuxStatus_t {
        NPF_uint16_t                          n_data;
        NPF_IfSONET_SDH_MuxStatusType_t   *statusinfo;
        };
```

2.1.3.2.4 SONET/SDH Adaptation Interface Status

```
/*
** SONET-SDH  Adaptation Interface Status
*/

typedef enum {

        NPF_unit32_t   AdapturStatus;
```

```
        NPF_IF_SONET_SDH_Adapt_NoDefect = 1,  /* No Defect */
        NPF_IF_SONET_SDH_Adapt_AIS = 2,       /* Adaptation (line) AIS */
        NPF_IF_SONET_SDH_Adapt_RDI = 3,       /* Adaptation RDI */
        NPF_IF_SONET_SDH_Adapt_SSF = 4 ,      /* Adaptation Server Signal Fail */
/*
*/
        NPF_IF_SONET_SDH_Adapt_LOP =  5   /* Adaptation Loss of Pointer */

} NPF_IfSONET_SDH_AdaptStatusType_t;

struct NPF_IfSONET_SDH_AdaptStatus{
        NPF_uint16_t                          n_data;
        NPF_IfSONET_SDH_AdaptStatusType_t   *statusinfo;
        };
```

2.1.3.2.5 SONET/SDH Mapping Interface Status

```
/*
** SONET-SDH  Mapping Interface Status
*/

typedef enum {

NPF_IF_SONET_SDH_Map_NoDefect = 1,    /* No Defect */
NPF_IF_SONET_SDH_Map_AIS = 2,         /* Path AIS */
NPF_IF_SONET_SDH_Map_RDI = 3,         /* Path RDI */
NPF_IF_SONET_SDH_Map_SSF = 4,         /* Path Server Signal Fail
/*
*/
NPF_IF_SONET_SDH_Map_EXC = 5,         /* Excessive Bit Error Rate */
NPF_IF_SONET_SDH_Map_DEG = 6,         /* Signal Degrade */
/*
*/
NPF_IF_SONET_SDH_Map_RFI = 7,         /* Path RFI */
/*
*/
NPF_IF_SONET_SDH_Map_UNEQ = 8,        /* Map Status Unequipped */
NPF_IF_SONET_SDH_Map_TIM = 9,         /* TTI Mismatch defect */
NPF_IF_SONET_SDH_Map_PLM = 10,        /* Payload Mismatch Defect(Corresp. to SLM )*/
NPF_IF_SONET_SDH_Map_LOM = 11         /* Loss of Multi-frame */

      } NPF_IfSONET_SDH_MapStatusType_t;

struct NPF_IfSONET_SDH_MapStatus{
        NPF_uint16_t                          n_data;
        NPF_IfSONET_SDH_MapStatusType_t   *statusinfo;
        };
```

2.1.4 Performance Monitoring Data Structures

SONET/SDH Performance Monitoring (PM) has two components: configuring and statistics.

2.1.4.1 SONET/SDH Performance Monitoring Configuration

2.1.4.1.1 SONET/SDH Medium SES Threshold Attribute

The Threshold attribute is used to determine the level after which ”alerts” have to be passed to the monitoring application or agent. These “alerts” are passed as NPF Interface Management Performance Monitoring Events.

```
/*
**  SONET-SDH  Medium Severely Errored Seconds (SES) Threshold
*/
typedef enum {
  NPF_IF_SONET_SDH_ThreshOther = 0,
                                    /* Other */
  NPF_IF_SONET_SDH_ThreshBellcore1991 = 1,
                                    /* Bellcore TR-NWT-000253, 1991... */
  NPF_IF_SONET_SDH_ThreshAnsi1993 = 2,
                                    /* ANSI T1.231, 1993, ...  */
  NPF_IF_SONET_SDH_ThreshItu1995 = 3,
                                    /* ITU-T G.826, 1995 ... */
  NPF_IF_SONET_SDH_ThreshAnsi1997 = 4
                                    /* ANSI T1.231, 1997 ... */
       } NPF_IfSONET_SDH_MediumSESthresholdSet_t;
```

SONET/SDH Performance Monitoring (PM) statistics are grouped as follows:

2.1.4.2 SONET/SDH Performance Monitoring Statistics

2.1.4.2.1 SONET/SDH Physical Medium Interface Statistics

There are no statistics for this type of SONET/SDH interface.

2.1.4.2.2 SONET Section, SDH Regenerator Section Interface Statistics

The statistics for this type of interface are:

```
/*
** SONET-SDH  Section Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_SectionStatistics{

       NPF_IfSONET_SDH_SectionStats_t              CurrentSectionStatistics;
       NPF_IfSONET_SDH_SectionIntervalStats_t      IntervalSectionStatistics;

       };
```

They are organized in two groups:

2.1.4.2.2.1 Current Section/Regenerator Section Statistics

- current (in the last 15 minutes)
    - section current Error Seconds (ES)

- section current Severely Errored Seconds (SES)
- section current Severely Errored Framing Seconds (SEFS)
- section current Coding Violation (CV)
- section current Background Block Error (BBE)

```
/*
 *      SONET-SDH  Section Statistics
 */
typedef struct {

      typedef struct {

      NPF_uint64_t     SectionCurrentESs;
                                  /* section current Error Seconds */
      NPF_uint64_t     SectionCurrentSESs;
                                  /* section current Severely Errored Seconds */
      NPF_uint64_t     SectionCurrentSEFSs
                                  /* section Severely Errored Framing Seconds */
      NPF_uint64_t     SectionCurrentCVs
                                  /* section current Coding Violation */
      NPF_uint64_t     SectionCurrentBBEs
                                  /* section current Background Block Error */

       } NPF_IfSONET_SDH_SectionStats _t;
```

2.1.4.2.2.2 Interval Section/Regenerator Section Statistics

- interval (in the previous 24 hours)
  - interval identifier
  - section interval Valid Data
  - section interval Error Seconds (ES)
  - section interval Severely Errored Seconds (SES)
  - section interval Severely Errored Framing Seconds (SEFS)
  - section interval Coding Violation (CV)
  - section current Background Block Error (BBE)

```
/*
 *      SONET-SDH  Interval Section Statistics
 */

typedef struct {

      NPF_uint32_t     SectionIntervalID;
                                  /* section Interval Number (0-96)*/
      NPF_boolean_t    SectionIntervalValidData;
                                  /* section Interval Valid Data */
      NPF_IfSONET_SDH_SectionStats_t  IntervalStats;

       } NPF_IfSONET_SDH_SectionIntervalStats_t;
```

2.1.4.2.3 SONET/SDH Multiplexing Interface PM Statistics

Multiplexing Interface (SONET Line or SDH Multiplexing Section) statistics are defined as two groups: local and far end: see details below. The Low Order and High Order Multiplexing Statistics are the same, the distinction between them is made based on the attributes of the Multiplexing Interface, i.e. low order or high order.

```
/*
** SONET-SDH  Multiplexing Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_MuxStatistics_t {

       NPF_IfSONET_SDH_MuxStats_t  LocalMuxStatistics;
       NPF_IfSONET_SDH_MuxStats_t  FarEndMuxStatistics;

       };


/*
 *     SONET-SDH  Mux Statistics
 */
typedef struct {

      typedef struct {

      NPF_uint64_t    MuxCurrentESs;
                                          /* Mux current Error Seconds */
      NPF_uint64_t    MuxCurrentSESs;
                                          /* Mux current Severely Errored Seconds */
      NPF_uint64_t    MuxCurrentCVs
                                          /* Mux current Coding Violation */
      NPF_uint64_t    MuxCurrentBBEs
                                          /* Mux current Background Block Error */
      NPF_uint64_t    MuxCurrentUAs
                                          /* Mux current Unavailable Seconds */

       } NPF_IfSONET_SDH_MuxBasicStats_t;


typedef struct {
      NPF_uint32_t    MuxIntervalID;
                                          /* Mux Interval Number (0-96)*/
      NPF_boolean_t   MuxIntervalValidData
                                          /* Mux Interval Valid Data */

      NPF_IfSONET_SDH_MuxBasictats_t   MuxIntervalStats;

       } NPF_IfSONET_SDH_MuxIntervalStats_t;


typedef struct {

       NPF_IfSONET_SDH_MuxBasicStats_t      CurrMuxStats;
       NPF_IfSONET_SDH_MuxIntervalStats_t   IntervalMuxStats;

       } NPF_IfSONET_SDH_MuxStats_t;
```

2.1.4.2.3.1 Local End Multiplexing (Line) Group Statistics

- current (in the last 15 minutes)
  - line current Error Seconds (ES)
  - line current Severely Errored Seconds (SES)
  - line current Coding Violation (CV)
  - line current Background Block Errors (BBE)
  - line current Unavailable Seconds (UAS)
- interval (in the previous 24 hours)
  - interval identifier

- line interval Error Seconds (ES)
- line interval Severely Errored Seconds (SES)
- line interval Coding Violation (CV)
- line interval Background Block Errors (BBE)
- line interval Unavailable Seconds (UAS)
- line interval Valid Data

2.1.4.2.3.2 Far End Multiplexing (Line) Group Statistics

- current (in the last 15 minutes)
  - far end line current Error Seconds (ES)
  - far end line current Severely Errored Seconds (SES)
  - far end line current Coding Violation (CV)
  - far end line current Background Block Errors (BBE)
  - far end line current Unavailable Seconds (UAS)
- interval (in the previous 24 hours)
  - far end interval identifier
  - far end line interval Error Seconds (ES)
  - far end line interval Severely Errored Seconds (SES)
  - far end line interval Coding Violation (CV)
  - far end line interval Background Block Errors (BBE)
  - far end line interval Unavailable Seconds (UAS)
  - far end line interval Valid Data

2.1.4.2.4 SONET/SDH Path (Mapping) Interface PM Statistics

Mapping Interface (Path) group statistics are defined as two groups: local and far end: see further below. The Low order and High order Mapping Statistics are the same, the distinction between them is made based on the attributes of the Mapping Interface, i.e. low order or high order.

```
/*
** SONET-SDH  Mapping Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_MapStatistics {
       NPF_IfSONET_SDH_PathStats_t  LocalPathStatistics;
       NPF_IfSONET_SDH_PathStats_t  FarEndPathStatistics;
       };


/*
 *     SONET-SDH  Path Statistics
 */
typedef struct {

      typedef struct {

      NPF_uint64_t    PathCurrentESs;
                                        /* Path current Error Seconds */
      NPF_uint64_t    PathCurrentSESs;
                                        /* Path current Severely Errored Seconds */
      NPF_uint64_t    PathCurrentCVs
                                        /* Path current Coding Violation */
      NPF_uint64_t    PathCurrentBBEs
                                        /* Path current Background Block Errors */
      NPF_uint64_t    PathCurrentUAs
```

```
                                        /* Path current Unavailable Seconds */

        } NPF_IfSONET_SDH_PathBasicStats_t;

      typedef struct {

      NPF_uint32_t     PathIntervalID;
                                        /* Path Interval Number (0-96)*/
      NPF_boolean_t    PathIntervalValidData;
                                        /* Path Interval Valid Data */
      NPF_IfSONET_SDH_PathBasicStats_t  MapIntervalStats;

        } NPF_IfSONET_SDH_PathIntervalStats_t;

} NPF_IfSONET_SDH_PathStats_t;

typedef struct {

        NPF_IfSONET_SDH_PathBasicStats_t      CurrPathStats;
        NPF_IfSONET_SDH_PathIntervalStats_t   IntervalPathStats;

        } NPF_IfSONET_SDH_PathStats_t;
```

2.1.4.2.4.1 Local End Mapping Interface (Path) Group Statistics

- current (in the last 15 minutes)
  - path current Error Seconds (ES)
  - path current Severely Errored Seconds (SES)
  - path current Coding Violation (CV)
  - path current Background Block Errors (BBE)
  - path current Unavailable Seconds (UAS)
- interval (in the previous 24 hours)
  - interval identifier
  - path interval Error Seconds (ES)
  - path interval Severely Errored Seconds (SES)
  - path interval Coding Violation (CV)
  - path interval Background Block Errors (BBE)
  - path interval Unavailable Seconds (UAS)
  - path interval Valid Data

2.1.4.2.4.2 Far End Mapping Interface (Path) Group Statistics

- current (in the last 15 minutes)
  - far end path current Error Seconds (ES)
  - far end path current Severely Errored Seconds (SES)
  - far end path current Coding Violation (CV)
  - far end path current Background Block Errors (BBE)
  - far end path current Unavailable Seconds (UAS)
- interval (in the previous 24 hours)
  - interval identifier
  - far end path interval Error Seconds (ES)
  - far end path interval Severely Errored Seconds (SES)
  - far end path interval Coding Violation (CV)

- far end path interval Background Block Errors (BBE)
- far end path interval Unavailable Seconds (UAS)
- far end path interval Valid Data

2.2 SONET/SDH Completion Callback Type Codes: NPF_IfCallbackType_t

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

```
/*
 *      Completion Callback Types for SONET-SDH
*/
#define NPF_IF_SONET_SDH_MediumStatusGet  ((NPF_IF_TYPE_SONET_SDH<<16)+1)
#define NPF_IF_SONET_SDH_MediumCurrStatusGet  ((NPF_IF_TYPE_SONET_SDH<<16)+2)
#define NPF_IF_SONET_SDH_MediumSESthresAttrSet  ((NPF_IF_TYPE_SONET_SDH<<16)+3)

/**/
#define NPF_IF_SONET_SDH_SectionFMConfigSet ((NPF_IF_TYPE_SONET_SDH<<16)+4)
#define NPF_IF_SONET_SDH_SectionStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+5)
#define NPF_IF_SONET_SDH_SectionStatisticsQuery ((NPF_IF_TYPE_SONET_SDH<<16)+6)

/**/
#define NPF_IF_SONET_SDH_MuxFMConfigSet ((NPF_IF_TYPE_SONET_SDH<<16)+7)
#define NPF_IF_SONET_SDH_MuxStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+8)
#define NPF_IF_SONET_SDH_MuxStatisticsQuery ((NPF_IF_TYPE_SONET_SDH<<16)+9)
/**/
#define NPF_IF_SONET_SDH_AdaptFMConfigSet ((NPF_IF_TYPE_SONET_SDH<<16)+10)
#define NPF_IF_SONET_SDH_AdaptStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+11)
/**/
#define NPF_IF_SONET_SDH_MapFMConfigSet ((NPF_IF_TYPE_SONET_SDH<<16)+12)
#define NPF_IF_SONET_SDH_MapStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+13)
#define NPF_IF_SONET_SDH_MapStatisticsQuery ((NPF_IF_TYPE_SONET_SDH<<16)+14)
```

The following table summarizes the type of callback, for each function in this specific Interface Management API:

Function Name	Type Code
NPF_IfSONET_SDH_MediumFMConfigSet	NPF_IF_SONET_SDH_MediumFMConfigSet
NPF_IfSONET_SDH_MediumStatusGet	NPF_IF_SONET_SDH_MediumStatusGet
NPF_IfSONET_SDH_MediumCurrStatusGet	NPF_IF_SONET_SDH_MediumCurrStatusGet
NPF_IfSONET_SDH_MediumSESthresAttrSet	NPF_IF_SONET_SDH_MediumSESthresAttrSet
NPF_IfSONET_SDH_SectionFMConfigSet	NPF_IF_SONET_SDH_SectionFMConfigGet
NPF_IfSONET_SDH_SectionStatusGet	NPF_IF_SONET_SDH_SectionStatusGet
NPF_IfSONET_SDH_SectionStatisticsQuery	NPF_IF_SONET_SDH_SectionStatisticsQuery
NPF_IfSONET_SDH_MuxFMConfigSet	NPF_IF_SONET_SDH_MapFMConfigSet
NPF_IfSONET_SDH_MuxStatusGet	NPF_IF_SONET_SDH_MapStatusGet

Function Name	Type Code
NPF_IfSONET_SDH_MuxStatisticsQuery	NPF_IF_SONET_SDH_MuxStatisticsQuery
NPF_IfSONET_SDH_AdaptFMConfigSet	NPF_IF_SONET_SDH_AdaptFMConfigSet
NPF_IfSONET_SDH_AdaptStatusGet	NPF_IF_SONET_SDH_AdaptStatusGet
NPF_IfSONET_SDH_MapFMConfigSet	NPF_IF_SONET_SDH_MapFMConfigSet
NPF_IfSONET_SDH_MapStatusGet	NPF_IF_SONET_SDH_MapStatusGet
NPF_IfSONET_SDH_MapStatisticsQuery	NPF_IF_SONET_SDH_MapStatisticsQuery

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 SONET/SDH interface type is supported, the following must be included in the union within the **NPF_IfAsyncResponse_t** structure:

```
/*
** To be inserted in 'npf_if_core.h'
*/
typedef struct NPF_IfSONET_SDH_MediumStatus NPF_IfSONET_SDH_MediumStatus_t;
typedef struct NPF_IfSONET_SDH_MediumCurrStatus NPF_IfSONET_SDH_MediumCurrStatus_t;
typedef struct NPF_IfSONET_SDH_SectionStatus NPF_IfSONET_SDH_SectionStatus_t;
typedef struct NPF_IfSONET_SDH_MuxStatus NPF_IfSONET_SDH_MuxStatus_t;
typedef struct NPF_IfSONET_SDH_AdaptStatus NPF_IfSONET_SDH_AdaptStatus_t;
typedef struct NPF_IfSONET_SDH_MapStatus NPF_IfSONET_SDH_MapStatus_t;
         /* Performance Monitoring Statistics */
typedef struct NPF_IfSONET_SDH_SectionStatistics NPF_IfSONET_SDH_SectionStatistics_t;
typedef struct NPF_IfSONET_SDH_MuxStatistics NPF_IfSONET_SDH_MuxStatistics_t;
typedef struct NPF_IfSONET_SDH_MapStatistics NPF_IfSONET_SDH_MapStatistics_t;


/*
** Asynchronous Response types for SONET-SDH interfaces
*/



      /* Fault Management Status */
      NPF_IfSONET_SDH_MediumStatus_t         *if_MediumStatus;
      NPF_IfSONET_SDH_MediumCurrStatus_t     *if_MediumCurrStatus;
      NPF_IfSONET_SDH_SectionStatus_t        *if_SectionStatus;
      NPF_IfSONET_SDH_MuxStatus_t            *if_MuxStatus;
      NPF_IfSONET_SDH_AdaptCurrentStatus_t   *if_AdaptStatus;
      NPF_IfSONET_SDH_MapCurrentStatus_t     *if_MapStatus;
      /* Performance Monitoring Statistics */
      NPF_IfSONET_SDH_SectionStatistics_t    *if_SectionStatistics;
      NPF_IfSONET_SDH_MuxStatistics_t *if_LineStatistic;
      NPF_IfSONET_SDH_MapStatistics_t *if_MapStatistics

/*
** End of SONET-SDH  Callback types
*/
```

The following table summarizes the type of callback, and information returned for each of the SONET/SDH Interface Management API function call:

Type Code	Structure Returned
NPF_IF_SONET_SDH_MediumFMConfigSet	Unused
NPF_IF_SONET_SDH_MediumStatusGet	NPF_IfSONET_SDH_MediumStatus_t
NPF_IF_SONET_SDH_MediumCurrStatusGet	NPF_IfSONET_SDH_MediumCurrStatus_t
NPF_IF_SONET_SDH_MediumSESthresAttrSet	Unused
NPF_IF_SONET_SDH_SectionFMConfigSet	Unused
NPF_IF_SONET_SDH_SectionStatusGet	NPF_IfSONET_SDH_SectionStatus_t
NPF_IF_SONET_SDH_SectionStatisticsQuery	NPF_IfSONET_SDH_SectionStatistics_t
NPF_IF_SONET_SDH_MuxFMConfigSet	Unused
NPF_IF_SONET_SDH_MuxStatusGet	NPF_IfSONET_SDH_MapStatus_t
NPF_IF_SONET_SDH_MuxStatisticsQuery	NPF_IfSONET_SDH_MapStatistics_t
NPF_IF_SONET_SDH_AdaptFMConfigSet	Unused
NPF_IF_SONET_SDH_AdaptStatusGet	NPF_IfSONET_SDH_AdaptStatus_t
NPF_IF_SONET_SDH_AdaptStatisticsQuery	Unused
NPF_IF_SONET_SDH_MapFMConfigSet	Unused
NPF_IF_SONET_SDH_MapStatusGet	NPF_IfSONET_SDH_MapStatus_t
NPF_IF_SONET_SDH_MapStatisticsQuery	NPF_IfSONET_SDH_MapStatistics_t

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

2.3 SONET/SDH Interface Management API Error Codes

The following codes are used as values of **`NPF_IfErrorType_t`**.

```
#define NPF_IF_E_SONET_SDH_CODE(code) (0x10000+(NPF_IF_TYPE_SONET_SDH<<8)+(code))
```

The following SONET/SDH Function Calls Error Codes are defined:

```
/*
** SONET-SDH  Error Codes
*/
/*
** Medium
*/


/* Invalid SONET-SDH  Medium Coding */
#define NPF_IF_E_INVALID_MEDIUM_CODING  NPF_IF_E_SONET_SDH_CODE(1)

/* Invalid SONET-SDH  Medium Line Type */
#define NPF_IF_E_INVALID_MEDIUM_LINETYPE NPF_IF_E_SONET_SDH_CODE(2)

/* Invalid SONET-SDH  Medium Loopback Configuration */
#define NPF_IF_E_INVALID_MEDIUM_LOOPBACK_CONFIG NPF_IF_E_SONET_SDH_CODE(3)

/* Invalid SONET-SDH  Medium SES Threshold Set */
#define NPF_IF_E_INVALID_MEDIUM_SES_THRESHOLDSET NPF_IF_E_SONET_SDH_CODE(4)

/* Invalid SONET-SDH  Medium Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_MEDIUM_FM_CONF NPF_IF_E_SONET_SDH_CODE(5)


/*
/*
```

```
** Section
*/

/* Invalid SONET-SDH  Section Interface Attribute */
#define NPF_IF_E_INVALID_SECTION_ATTR NPF_IF_E_SONET_SDH_CODE(6)


/* Invalid SONET-SDH  SECTION Type */
#define NPF_IF_E_INVALID_SECTION _TYPE NPF_IF_E_SONET_SDH_CODE(7)


/* Invalid SONET-SDH  Section Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_SECTION_FM_CONF NPF_IF_E_SONET_SDH_CODE(8)


/*
** Multiplexing
*/
/* Invalid SONET-SDH  Multiplexing Interface Attribute */
#define NPF_IF_E_INVALID_MUX_ATTR NPF_IF_E_SONET_SDH_CODE(9)

/* Invalid SONET-SDH  Multiplexing Interface Configuration Parameter */
#define NPF_IF_E_INVALID_MUX_FM_CONF NPF_IF_E_SONET_SDH_CODE(10)

/*
** Adaptation
*/


/* Invalid SONET-SDH  Adaptation Interface Attribute */
#define NPF_IF_E_INVALID_ADAPT_ATTR NPF_IF_E_SONET_SDH_CODE(11)

/* Invalid SONET-SDH  Adaptation Interface Configuration Parameter */
#define NPF_IF_E_INVALID_ADAPT_FM_CONF NPF_IF_E_SONET_SDH_CODE(12)

/*
** Mapping
*/

/* Invalid SONET-SDH  MappingInterface Attribute */
#define NPF_IF_E_INVALID_MAP_ATTR NPF_IF_E_SONET_SDH_CODE(13)


/* Invalid SONET-SDH  Mapping Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_MAP_FM_CONFIG NPF_IF_E_SONET_SDH_CODE(14)

/*
** All SONET-SDH  parent-child relationship
*/

/* Invalid SONET-SDH  Interface Binding */
#define NPF_IF_E_INVALID_BINDING NPF_IF_E_SONET_SDH_CODE(15)

/*
** End of SONET-SDH  Error Codes
*/
```

2.4 SONET/SDH Interface Management API Events

The SONET/SDH events are of two categories: Fault Management events and Performance Monitoring events. The SONET/SDH events must be added to the “core” document definition of events.

2.4.1 SONET/SDH Fault Management Events

```
/*
** SONET-SDH  Interface Management Fault Management Events
*/

/*
** SONET-SDH  Physical Medium Events
*/
#define NPF_IF_SONET_SDHMedium_LOF ((NPF_IF_TYPE_SONET_SDH<<16)+1)

                                    /* Medium Loss of Frame Failure */
#define NPF_IF_SONET_SDH_Medium_LOS ((NPF_IF_TYPE_SONET_SDH<<16)+1)
                                    /* Medium Loss of Signal */
/*
** SONET-SDH  Section Events
*/
#define NPF_IF_SONET_SDH_Section_TIM ((NPF_IF_TYPE_SONET_SDH<<16)+1)
                                    /* Section Loss of Frame Failure */
#define NPF_IF_SONET_SDH_Section_SSF ((NPF_TYPE_SONET_SDH<<16)+ 13)
                                    /* Section Loss of Signal */

/*
**  SONET-SDH  Multiplexing Interface (line Multiplexing Section) Events
*/
#define NPF_IF_SONET_SDH_Line_AIS ((NPF_IF_TYPE_SONET_SDH<<16)+1)
                                /* Multiplexing Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Line_RDI ((NPF_TYPE_SONET_SDH<<16)+ 15)
                                /* Multiplexing Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Line_SSF ((NPF_TYPE_SONET_SDH<<16)+ 16)
                                /* Multiplexing Interface Server Signal Failure */
       /*
       */
#define NPF_IF_SONET_SDH_Line_EXC ((NPF_TYPE_SONET_SDH<<16)+ 17)
                                /* Multiplexing Interface Excessive Bit Error Rate */
#define NPF_IF_SONET_SDH_Line_DEC ((NPF_TYPE_SONET_SDH<<16)+ 18)
                                /* Multiplexing Interface Signal Degrade */

/*
** SONET-SDH  Adaptation Interface Events
*/
#define NPF_IF_SONET_SDH_Adapt_AIS ((NPF_TYPE_SONET_SDH<<16)+ 19)
                                    /* Adaptation Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Adapt_RDI ((NPF_TYPE_SONET_SDH<<16)+ 20)
                                    /* Adaptation Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Adapt_SSF ((NPF_TYPE_SONET_SDH<<16)+ 21)
                                    /* Adaptation Interface Server Signal Failure */
#define NPF_IF_SONET_SDH_Adapt_LOP ((NPF_TYPE_SONET_SDH<<16)+ 22)
                                    /* Adaptation Interface Loss of Pointer */
/*
** SONET-SDH  Mapping Interface Events
*/
#define NPF_IF_SONET_SDH_Map_AIS ((NPF_TYPE_SONET_SDH<<16)+ 23)
                                    /* Mapping Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Map_RDI ((NPF_TYPE_SONET_SDH<<16)+ 24)
                                    /* Mapping Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Map_SSF ((NPF_TYPE_SONET_SDH<<16)+ 25)
                                    /* Mapping Interface Server Signal Failure */
       /*
       */
#define NPF_IF_SONET_SDH_Map_EXC ((NPF_TYPE_SONET_SDH<<16)+ 26)
                                    /* Mapping Interface Excessive Bit Error Rate */
#define NPF_IF_SONET_SDH_Map_DEC ((NPF_TYPE_SONET_SDH<<16)+ 27)
                                    /* Mapping Interface Signal Degrade */
       /*
       */
#define NPF_IF_SONET_SDH_Map_RFI ((NPF_TYPE_SONET_SDH<<16)+ 28)
                                    /* Mapping Interface Remote Failure Indication */
```

```
        /*
        */
#define NPF_IF_SONET_SDH_Map_UNEQ ((NPF_TYPE_SONET_SDH<<16)+ 29)
                                        /* Mapping Interface Status Unequipped */
#define NPF_IF_SONET_SDH_Map_TIM ((NPF_TYPE_SONET_SDH<<16)+ 30)
                                        /* Mapping Interface TTI Mismatch Defect */
#define NPF_IF_SONET_SDH_Map_PLM ((NPF_TYPE_SONET_SDH<<16)+ 31)
                                        /* Mapping Interface Payload Mismatch Defect */
#define NPF_IF_SONET_SDH_Map_LOM ((NPF_TYPE_SONET_SDH<<16)+ 32)
                                        /* Mapping Interface Loss of Multi-Frame */
```

2.4.2 SONET/SDH Performance Monitoring Events

The SONET/SDH Performance Monitoring events are triggered by SONET/SDH Performance Monitoring alert notifications sent by the SONET/SDH layers upon detecting a crossing of the threshold level.

```
/*
**  SONET-SDH  Performance Monitoring Events
*/


#define NPF_IF_SONET_SDH_ES ((NPF_TYPE_SONET_SDH<<16) + 100 + 4)
        /* Errored Seconds */
#define NPF_IF_SONET_SDH_SES ((NPF_TYPE_SONET_SDH<<16) + 100 + 5)
        /* Severe Errored Seconds */
#define NPF_IF_SONET_SDH_SEFS ((NPF_TYPE_SONET_SDH<<16) + 100 + 6)
        /* Severely Errored Framing Seconds */
#define NPF_IF_SONET_SDH_CV ((NPF_TYPE_SONET_SDH<<16) + 100 + 7)
        /* Coding Violation */
#define NPF_IF_SONET_SDH_US ((NPF_TYPE_SONET_SDH<<16) + 100 +8)
        /* Unavailable Seconds */
```

2.4.3 SONET/SDH Event Notification

SONET/SDH Event notification will be performned through the existing NPF Event data structures which are defined in the IM Core API document.

3 SONET/SDH Interface Management API Function Calls

The following existing NPF Interface Management API generic function calls can be used with SONET/SDH interfaces:

NPF_IfCreate()
NPF_IfDelete()
NPF_IfCreateAndSet()
NPF_IfBind()
NPF_IfUnbind()

NPF_IfGenericStatsGet()
NPF_IfAttrSet()
NPF_IfAttrGet()
NPF_IfAdminStatus()
NPF_IfEnable()
NPF_IfDisable()
NPF_IfOperStatusGet()

Specific SONET/SDH error codes may be returned by the following generic function calls: NPF_IfAttrSet, and NPF_IfCreateAndSet().

Enabling/disabling the asynchronous completion of the SONET/SDH function calls is controlled by the existing NPF_ifRegister(), and NPF_ifDeregister() function calls.

Enabling/disabling the event notification mechanism for the SONET/SDH interfaces is controlled by the existing NPF_ifEventRegister(), and NPF_ifEventDeregister() function calls.

3.1 Generic Function Calls

3.1.1 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.

SONET/SDH Interface Errors:

- **NPF_IF_E_INVALID_MEDIUM_CODING**: Invalid SONET/SDH medium coding.
- **NPF_IF_E_INVALID_MEDIUM_LINE_TYPE**: Invalid SONET/SDH medium line type.
- **NPF_IF_E_INVALID_MEDUM_LOOPBACK_CONFIGURATION:** Invalid medium loopback configuration
- **NPF_IF_E_INVALID_SECTION_ATTR :** Invalid SONET/SDH Section Interface Attribute.
- **NPF_IF_E_INVALID_MUX_ATTR:** Invalid SONET/SDH Multiplexing Interface Attribute
- **NPF_IF_E_INVALID_ADAPT_ATTR:** Invalid SONET/SDH Adaptation Interface Attribute
- **NPF_IF_E_INVALID_MAP_ATTR:** Invalid SONET/SDH Mapping 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.

3.1.2 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.

SONET/SDH Interface Errors:

- **NPF_IF_E_INVALID_MEDIUM_CODING**: Invalid SONET/SDH medium coding.
- **NPF_IF_E_INVALID_MEDIUM_LINE_TYPE**: Invalid SONET/SDH medium line type.
- **NPF_IF_E_INVALID_MEDUM_LOOPBACK_CONFIGURATION:** Invalid medium loopback configuration
- **NPF_IF_E_INVALID_SECTION_ATTR :** Invalid SONET/SDH Section Interface Attribute.
- **NPF_IF_E_INVALID_MUX_ATTR:** Invalid SONET/SDH Multiplexing Interface Attribute
- **NPF_IF_E_INVALID_ADAPT_ATTR:** Invalid SONET/SDH Adaptation Interface Attribute
- **NPF_IF_E_INVALID_MAP_ATTR:** Invalid SONET/SDH Mapping 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.

3.1.3 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 SONET/SDH 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 SONET/SDH parent interface to another SONET/SDH interface means that the parent interface is a lower layer in the SONET/SDH hierarchy, while the child is a higher layer in the SONET/SDH hierarchy.
- Not all combinations of SONET/SDH parent-child interfaces are valid. An implementation MUST return a binding error if the SONET/SDH binding combination is invalid.
- The following SONET/SDH interface bindings are valid (see Figure 2-1, Figure 2-2, Figure 2-3, Figure 2-4, and Figure 2-6):
    - SONET/SDH Medium interface parent to one or more SONET Section interfaces
    - SONET/SDH Medium interface parent to one or more SDH Regenerator Section interfaces
    - SONET Section interface parent to one or more SONET STS Multiplexing interfaces
    - SDH Regenarator Section interface parent to one or more SDH High Order Multiplexing Section interfaces

  - SONET STS Multiplexing interface parent to one or more SONET STS Adaptation interfaces, or one or more SONET VT Adaptaton interfaces.
  - SDH High Order Multiplexing Section interface parent to one or more SDH High Order Adaptation interfaces, or one or more SDH Low Order Adaptation interfaces.
  - SONET STS or VT Adaptation interface parent to one SONET STS SPE respectively VT Mapping interface.
  - SDH High Order or Low Order Adaptation interface parent to one SDH High Order or Low Order Mapping interface
  - SONET STS Mapping interface parent to one or more VT Group Multiplexing interfaces
  - SDH Mapping interface parent to one or more Low Order Multiplexing 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.

SONET/SDH Interface Errors:

- **`NPF_IF_E_INVALID_BINDING:`** Invalid SONET/SDH 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.

3.2 SONET/SDH Interface Attribute Setting Function Calls

SONET, or SDH Interface attributes can be set with the Interface Management Generic function calls.

3.3 SONET/SDH Fault Management Function Calls

3.3.1 NPF_IfSONET_SDH_MediumFMConfigSet

Function to Set the Fault Management Configuration parameters of a SONET/SDH Medium Interface.

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MediumFMConfigSet (
        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_IfSONET_SDH_Medium_FMConfig_t
                                      *if_SONET_SDH_MediumFMConfArray);
```

Description of function

This function sets the Fault Management parameters of one or more SONET/SDH Medium type interfaces. The ‘if_HandleArray’ and ‘if_SONET_SDH_MediumFMConfArray’ arrays must both contain the same number of

entries, equal to the value of 'n_handles'. The Fault Management Configuration parameters of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_MediumFMConfArray: pointer to an array of SONET/SDH Medium Interface Fault Management Configuration parameters structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH interface.
- **`NPF_IF_E_INVALID_MEDIUM_FM_CONF`**: Invalid SONET/SDH Medium Interface Fault Management Configuration parameters.

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

3.3.2 NPF_IfSONET_SDH_SectionFMConfigSet

Function to set Fault Management Configuration parameters to a SONET Section or SDH Regenerator Section Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_SectionFMConfigSet (
       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_IfSONET_SDH_Section_FMConf_t
                                     *if_SONET_SDH_SectionFMConfArray);
```

Description of function

This function sets the Fault Management Configuration parameters of one or more SONET Section or SDH Regenerator Section interfaces. The 'if_HandleArray' and 'if_SONET_SDH_SectionFMConfArray' arrays must both contain the same number of entries, equal to the value of 'n_handles'. The Fault Management Configuration parameters of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_SectionFMConfArray: pointer to an array of SONET/SDH Section Interface Fault Management Configuration parameters structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH Section interface.
- **`NPF_IF_E_INVALID_SECTION_FM_CONF:`** Invalid SONET/SDH Section Interface Fault Management Configuration parameters.

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.

3.3.3 NPF_IfSONET_SDH_MuxFMConfigSet

Function to set Fault Management Configuration parameters to a SONET/SDH Multiplexing Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MuxFMConfigSet (
        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_IfSONET_SDH_Mux_FMConf_t
                              *if_SONET_SDH_MuxFMConfArray);
```

Description of function

This function sets the Fault Management Configuration parameters of one or more SONET Multiplexing interfaces. The 'if_HandleArray' and 'if_SONET_SDH_MuxFMConfArray' arrays must both contain the same number of entries, equal to the value of 'n_handles'. The Fault Management Configuration parameters of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_MuxFMConfArray: pointer to an array of SONET/SDH Multiplexing Interface Fault Management Configuration parameters structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH Section interface.
- **`NPF_IF_E_INVALID_MUX_FM_CONF:`** Invalid SONET/SDH Multiplexing Interface Fault Management Configuration parameters.

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.

3.3.4 NPF_IfSONET_SDH_AdaptFMConfigSet

Function to set Fault Management Configuration parameters to a SONET Adaptation Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_AdaptFMConfigSet (
        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_IfSONET_SDH_Adapt_FMConf_t
                                      *if_SONET_SDH_AdaptFMConfArray);
```

Description of function

This function sets the Fault Management Configuration parameters of one or more SONET Adaptation interfaces. The 'if_HandleArray' and 'if_SONET_SDH_AdaptFMConfArray' arrays must both contain the same number of entries, equal to the value of 'n_handles'. The Fault Management Configuration parameters of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_AdaptFMConfArray: pointer to an array of SONET/SDH Adaptatoin Interface Fault Management Configuration parameters structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH Section interface.
- **`NPF_IF_E_INVALID_ADAPT_FM_CONF:`** Invalid SONET/SDH Adaptation Interface Fault Management Configuration parameters.

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.

3.3.5 NPF_IfSONET_SDH_MapFMConfigSet

Function to set Fault Management Configuration parameters to a SONET Mapping Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MapFMConfigSet (
        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_IfSONET_SDH_Map_FMConf_t
                                      *if_SONET_SDH_MapFMConfArray);
```

Description of function

This function sets the Fault Management Configuration parameters of one or more SONET Mapping interfaces. The 'if_HandleArray' and 'if_SONET_SDH_MapFMConfArray' arrays must both contain the same number of entries, equal to the value of 'n_handles'. The Fault Management Configuration parameters of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_MapFMConfArray: pointer to an array of SONET/SDH Mapping Interface Fault Management Configuration parameters structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH Section interface.
- **`NPF_IF_E_INVALID_MAP_FM_CONF:`** Invalid SONET/SDH Mapping Interface Fault Management Configuration parameters.

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.

3.3.6 NPF_IfSONET_SDH_MediumStatusGet

Function to Get Medium Status for a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MediumStatusGet(
        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 SONET/SDH Medium 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 SONET/SDH Medium.

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 SONET/SDH Medium 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 SONET/SDH 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 SONET/SDH Medium Status structure.

3.3.7 NPF_IfSONET_SDH_MediumCurrStatusGet

Function to Get Medium Current Status for a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MediumCurrStatusGet(
        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 SONET/SDH Medium Current 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 SONET/SDH Medium

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 SONET/SDH 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 SONET/SDH 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 SONET/SDH Medium Status structure.

3.3.8 NPF_IfSONET_SDH_SectionStatusGet

Function to Get Section Status for a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_SectionStatusGet(
        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, the SONET/SDH Section status.

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 SONET/SDH Section

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 SONET/SDH Section 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 SONET/SDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a SONET/SDH Section status.

3.3.9 NPF_IfSONET_SDH_MuxStatusGet

Function to Get Line Status for a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MuxStatusGet(
        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, the SONET/SDH Line status.

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 SONET/SDH Multiplexing

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 SONET/SDH Line 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 SONET/SDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a SONET/SDH Line status.

3.3.10 NPF_IfSONET_SDH_AdaptStatusGet

Function to Get Path Status for a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_AdaptStatusGet(
        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, the status of a SONET/SDH Adaptation interface.

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 SONET/SDH Adaptation

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 SONET/SDH Adaptation 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 SONET/SDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a SONET/SDH Path status.

3.3.11 NPF_IfSONET_SDH_MapStatusGet

Function to Get Status of a SONET/SDH Mapping Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MapStatusGet(
        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, the SONET/SDH Mapping Interface status.

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 SONET/SDH Mapping.

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 SONET/SDH Mapping 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 SONET/SDH interface and a success code or a possible error code. If the error code indicates success, the union in the response structure contains a SONET/SDH Mapping Interface status.

3.4 SONET/SDH Performance Monitoring Function Calls

3.4.1 NPF_IfSONET_SDH_MediumSESthreshSet

Function to Set the Medium SES Threshold Set attribute of a SONET/SDH Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MediumSESthreshAttrSet (
        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_IfSONET_SDH_MediumSESthresholdSet_t
                       *if_SONET_SDH_MediumSESthreshSetArray);
```

Description of function

This function sets the Medium SES Threshold Set attribute of one or more SONET/SDH interfaces. The 'if_HandleArray' and 'if_SONET_SDH_MediumSESthesShSetArray' arrays must both contain the same number of entries, equal to the value of 'n_handles'. The Medium Loopback Configuration attributes of each interface is set from a *different* element of the attributes array.

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 the Attribute for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- If_SONET_SDH_MediumSESthresholdSetArray: pointer to an array of SONET/SDH Severly Errored Seconds (SES) Threshold Set structures.

Output Parameters

None

Asynchronous Error Codes

- **`NPF_NO_ERROR`**: Operation successful.
- **`NPF_IF_E_INVALID_HANDLE`**: An **`if_Handle`** is null or invalid, or is not a SONET/SDH interface.
- **`NPF_IF_E_INVALID_MEDIUM_SES_TRESHOLDSET:`** Invalid SONET/SDH SES threshold set.

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.

3.4.2 NPF_IfSONET_SDH_SectionStatisticsQuery

Function to Get Statistics for a SONET/SDH Section Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_SectionStatisticsQuery(
       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 SONET/SDH Section interface 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_IfSONET_SDH_SectionStatistics_t structure for that interface.

3.4.3 NPF_IfSONET_SDH_MuxStatisticsQuery

Function to Get Statistics of a SONET/SDH Multiplexing Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MuxStatisticsQuery(
        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 SONET/SDH Multiplexing 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_IfSONET_SDH_MuxStatistics_t` structure for that interface.

3.4.4 NPF_IfSONET_SDH_MapStatisticsQuery

Function to Get Statistics of a SONET/SDH Mapping Interface

Syntax

```
NPF_error_t  NPF_IfSONET_SDH_MapStatisticsQuery(
        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 SONET/SDH Mapping interface 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_IfSONET_SDH_MapStatistics_t` structure for that interface.

4 Summary

4.1 Symmary of API Functions

The following is a summary table of the SONET/SDH Function Calls:

Function Name	Required?
NPF_IfAttrSet()	Yes
NPF_IfCreateAndSet()	Yes
NPF_IfSONET_SDH_SectionFMConfigSet()	Yes
NPF_IfSONET_SDH_MuxFMConfigSet()	Yes
NPF_IfSONET_SDH_AdaptFMConfigSet()	Yes
NPF_IfSONET_SDH_MapFMConfigSet()	Yes
NPF_IfSONET_SDH_MediumStatusGet ()	Yes
NPF_IfSONET_SDH_MediumCurrStatusGet ()	Yes
NPF_IfSONET_SDH_MediumCurrStatusGet ()	Yes
NPF_IfSONET_SDH_MediumStatusGet ()	Yes
NPF_IfSONET_SDH_SectionStatusGet ()	Yes
NPF_IfSONET_SDH_MuxStatusGet ()	Yes
NPF_IfSONET_SDH_AdaptStatusGet ()	Yes
NPF_IfSONET_SDH_MapStatusGet ()	Yes
NPF_IfSONET_SDH_MediumSESthresholdAttrSet()	Yes
NPF_IfSONET_SDH_SectionStatisticsQuery ()	Yes
NPF_IfSONET_SDH_MuxStatisticsQuery ()	Yes
NPF_IfSONET_SDH_MapStatisticsQuery ()	Yes

Table 4-1 Summary of SONET/SDH Function Calls

4.2 Summary of API Functions and Input Data structures

The following table summarizes the Function Calls and the Data Structures used as Input Parameters:

Function Name	Data Structure
NPF_IfAttrSet()	NPF_IfGeneric_t
NPF_IfCreateandSet()	NPF_IfGeneric_t
NPF_IfSONET_SDH_MediumFMConfigSet	NPF_IfSONET_SDH_Medium_FMConfig_t
NPF_IfSONET_SDH_SectionFMConfigSet	NPF_IF_SONET_SDH_Section_MConfigGet
NPF_IfSONET_SDH_MuxFMConfigSet	NPF_IF_SONET_SDH_Mux_FMConf_t
NPF_IfSONET_SDH_AdaptFMConfigSet	NPF_IF_SONET_SDH_Adapt_FMConf_t
NPF_IfSONET_SDH_MapFMConfigSet	NPF_IF_SONET_SDH_Map_FMConfig_t
NPF_IfSONET_SDH_MediumSESthresAttrSet	NPF_IfSONET_SDH_MediumSESthresholdSet_t

Table 4-2 Table of Function Names and Input Data Structures

5 References

- NPF Software API Conventions IA
- [NPF2002.471.04] NPF Interface Management API IA revision 2.0
- IETF - RFC3592 :Definitions of Managed Objects for the SONET/SDH Interface Type

- ITU-T – G.707
- ITU-T - G.783
- ATIS/T1X1 – T1.105,
- ATIS/T1X1 - T1.231
- TelCordia – GR-253
- ETSI EN 300 417-1-1

6 Revision History

V00	03/11/2004	Merge individual sections drafts into this SONET/SDH API Initial draft.
V01	03/18/2004	Make sure .h file in Appendix compiles without errors. Added subsection in Introduction.
V02	03/19/2004	Make some corrections to the text
V03	05/17/2004	Update and prepare contribution for NPF Interim Meeting
V04	07/09/2004	Update draft to new IM API model . Add support for VCAT, VCG, and APS
V05	07/10/2004	Add more text
V06	07/10/2004	Make corrections
V07	08/20/2004	Make corrections to the ‘sonet-sdh.h’ file, and code sequences in the specification.
V08	08/30/2004	Eliminate compile errors from ‘sonet-sdh.h’ file (Appendix A).
V09	09/02/2004	Eliminate all compile errors from ‘sonet-sdh.h’ file (Appendix A).
V10	10/10/04	Straw Ballot Comment Resolution

Appendix A Header File: NPF_IF_SONET/SDH.h

```
/* NPF_IF_SONET_SDH.h */

/*
**  This header file defines typedefs, constants, and functions
**  that apply to the NPF SONET/SDH 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_SONET_SDH_H_
#define __NPF_IF_SONET_SDH_H_

#ifdef __cplusplus
extern "C"  {
#endif


/*
**  *** SONET/SDH Definitions
*/



/*
** --- SONET/SDH Fault Management Data structures
*/
/*
** +++ SONET/SDH Fault Management Configuration Data structures
*/

/*
** Configuration for SONET/SDH Trail Termination Functions
*/


typedef enum {
  NPF_IF_SONET_SDH_NonMonitoringState = 0,
/* Interface in non monitoring mode
** for Trail Termination functions
** (Tpmode = NMON)
*/
  NPF_IF_SONET_SDH_MonitoringState = 1
/* Interface in monitoring mode
** for Trail Termination functions
** (Tpmode = MON)
*/
} NPF_IfSONET_SDH_TPMode_t;
```

```
/*
** Configuration SONET/SDH Server Signal Failure reporting
*/

typedef enum {
NPF_IF_SONET_SDH_SSF_ReportDisable = 0,
/* Interface configured not to report
** Server Signal Failure
*/
NPF_IF_SONET_SDH_SSF_ReportEnable = 1
/* Interface configured to report SSF */
} NPF_IfSONET_SDH_SSF_ReportMode_t;

/*
**  Configuration of SONET/SDH Alarm Indication Signal (AIS) reporting
Mode
*/

typedef enum {
NPF_IF_SONET_SDH_AIS_ReportDisable = 0,
/* Interface configured not to report AIS signal */
NPF_IF_SONET_SDH_AIS_ReportEnable = 1
/* Interface configured to report AIS signal */
} NPF_IfSONET_SDH_AIS_ReportMode_t;

/*
** Configuration SONET/SDH Remote Defect Information reporting
*/
typedef enum {
NPF_IF_SONET_SDH_RDI_ReportDisable = 0,
/* Inteface configured not to report Remote Defect Information */
NPF_IF_SONET_SDH_RDI_ReportEnable = 1
/* Interface configured to report Remote Defect Information */
} NPF_IfSONET_SDH_RDI_ReportMode_t;


/*
** SONET/SDH Medium Interface Fault Management Configuration Parameters
*/


typedef struct {
 NPF_IfSONET_SDH_TPMode_t TPMode;     /* Trail Termination monitoring
*/
 NPF_IfSONET_SDH_SSF_ReportMode_t SSF_ReportMode; /* Server Signal Fail
*/
 NPF_IfSONET_SDH_AIS_ReportMode_t AIS_ReportMode; /* Alarm Indication
Signal */
 NPF_IfSONET_SDH_RDI_ReportMode_t RDI_ReportMode; /* Remote Defect
Information */
    } NPF_IfSONET_SDH_MediumFMConfig_t;


/*
```

```
** Configuration SONET/SDH Server Signal Failure reporting
*/

typedef enum {
NPF_IF_SONET_SDH_SECTION_SSF_ReportDisable = 0,
/* Interface configured not to report
** Server Signal Failure
*/
NPF_IF_SONET_SDH_SECTION_SSF_ReportEnable = 1
/* Interface configured to report SSF */
} NPF_IfSONET_SDH_Section_SSF_RepMode_t;



typedef enum {
NPF_IF_SONET_SDH_SECTION_AIS_ReportDisable = 0,
/* Interface configured not to report AIS signal */
NPF_IF_SONET_SDH_SECTION_AIS_ReportEnable = 1
/* Interface configured to report AIS signal */
} NPF_IfSONET_SDH_Section_AIS_RepMode_t;


/*
** Configuration SONET/SDH Remote Defect Information reporting
*/
typedef enum {
NPF_IF_SONET_SDH_SECTION_RDI_ReportDisable = 0,
/* Inteface configured not to report Remote Defect Information */
NPF_IF_SONET_SDH_SECTION_RDI_ReportEnable = 1
/* Interface configured to report Remote Defect Information */
} NPF_IfSONET_SDH_Section_RDI_RepMode_t;

typedef enum {
NPF_IF_SONET_SDH_SECTION_TIMdis = 0, /* RS Section TIM processing
disabled */
NPF_IF_SONET_SDH_SECTION_TIMen = 1 /* RS  Section TIM processing
enabled */
} NPF_IfSONET_SDH_Section_TIMCfgSet_t ;

typedef enum {
NPF_IF_SONET_SDH_SECTION_TTI_AISdis = 0,
/* AIS insertion on Regenerator Section TTI mismtach disabled */
NPF_IF_SONET_SDH_SECTION_TTI_AISen = 1
/* AIS insertion on Regenerator Section TTI mismtach disabled */
} NPF_IfSONET_SDH_Section_TTI_AISCfgSet_t;

typedef enum {
NPF_IF_SONET_SDH_SECTION_TxFECOff = 0, /* FEC Calculation on Tx not
enabled */
NPF_IF_SONET_SDH_SECTION_TxFECOn = 1   /* FEC Calculation on Tx Enabled
*/
} NPF_IfSONET_SDH_Section_Tx_FECCfgSet_t;

typedef enum {
NPF_IF_SONET_SDH_SECTION_RxFECOff = 0, /* FEC Calculation on Rx not
enabled */
```

```
NPF_IF_SONET_SDH_SECTION_RxFECOn = 1   /* FEC Calculation on Rx Enabled
*/
} NPF_IfSONET_SDH_Section_Rx_FECCfgSet_t;

typedef enum {
NPF_IF_SONET_SDH_SECTION_TxDelayOff = 0,    /* Tx Delay Buffers not
used */
NPF_IF_SONET_SDH_SECTION_TxDelay = 1        /* Tx Delay Buffers used */
} NPF_IfSONET_SDH_Section_Tx_DelayCfgSet_t;


typedef enum {
NPF_IF_SONET_SDH_SECTION_RxDelayOff = 0,    /* Rx Delay Buffers not
used */
NPF_IF_SONET_SDH_SECTION_RxDelay = 1        /* Rx Delay Buffers used */
} NPF_IfSONET_SDH_Section_Rx_DelayCfgSet_t;

/*
** SONET-SDH Section Fault Management Configuration
*/
typedef struct {
      NPF_IfSONET_SDH_Section_AIS_RepMode_t  AIS_SecReportMode;
/* Alarm Indication Signal */
    NPF_IfSONET_SDH_Section_RDI_RepMode_t  RDI_SecReportMode;
/* Remote Defect Information */
    NPF_IfSONET_SDH_Section_SSF_RepMode_t  SSF_SecReportMode;
/* Server Signal Fail */
    NPF_IfSONET_SDH_Section_TIMCfgSet_t    Section_TIMCfgSet;
              /* Section-RS TIM processing enabled/disabled */
    NPF_IfSONET_SDH_Section_TTI_AISCfgSet_t     Section_TTIAISCfgSet;
/* AIS Insertion on Section-RS TTI and AIS enabled/disabled */
/*
** FEC and Delay buffers are to be used only from OC-48 and up.
*/
   NPF_IfSONET_SDH_Section_Tx_FECCfgSet_t  Section_Tx_FECCfgSet_t;
   NPF_IfSONET_SDH_Section_Rx_FECCfgSet_t      Section_Rx_FECCfgSet_t;
   NPF_IfSONET_SDH_Section_Tx_DelayCfgSet_t    Section_Tx_DelCfgSet_t;
   NPF_IfSONET_SDH_Section_Rx_DelayCfgSet_t    Section_Rx_DelCfgSet_t;

}NPF_IfSONET_SDH_Section_FMConf_t;


/*
** SONET-SDH Multiplexing Fault Management Configuration
*/




/*
** Configuration SONET/SDH Server Signal Failure reporting
*/

typedef enum {
NPF_IF_SONET_SDH_MUX_SSF_ReportDisable = 0,
/* Interface configured not to report
```

```
** Server Signal Failure
*/
NPF_IF_SONET_SDH_MUX_SSF_ReportEnable = 1
/* Interface configured to report SSF */
} NPF_IfSONET_SDH_Mux_SSF_RepMode_t;
/*
**  Configuration of SONET/SDH Alarm Indication Signal (AIS) reporting
Mode
*/

typedef enum {
NPF_IF_SONET_SDH_MUX_AIS_ReportDisable = 0,
/* Interface configured not to report AIS signal */
NPF_IF_SONET_SDH_MUX_AIS_ReportEnable = 1
/* Interface configured to report AIS signal */
} NPF_IfSONET_SDH_Mux_AIS_RepMode_t;

/*
** Configuration SONET/SDH Remote Defect Information reporting
*/
typedef enum {
NPF_IF_SONET_SDH_MUX_RDI_ReportDisable = 0,
/* Inteface configured not to report Remote Defect Information */
NPF_IF_SONET_SDH_MUX_RDI_ReportEnable = 1
/* Interface configured to report Remote Defect Information */
} NPF_IfSONET_SDH_Mux_RDI_RepMode_t;
/*
**  SONET-SDH Mapping Interface Signal Degeneration Threshold
*/
typedef enum {

  NPF_IF_SONET_SDH_MUX_ThreshBursty = 5,

  NPF_IF_SONET_SDH_MUX_ThreshPoisson = 6

} NPF_IfSONET_SDH_MuxThresh_t;


/*
** Multiplexing FM structure
*/

typedef struct {

NPF_IfSONET_SDH_Mux_AIS_RepMode_t       AIS_MuxReportMode;
/* Alarm Indication Signal */
NPF_IfSONET_SDH_Mux_RDI_RepMode_t       RDI_MuxReportMode;
/* Remote Defect Information */
NPF_IfSONET_SDH_Mux_SSF_RepMode_t       SSF_MuxReportMode;
/* Server Signal Fail */
NPF_IfSONET_SDH_MuxThresh_t             SDH_MuxThresh;
/* Threshold */
}NPF_IfSONET_SDH_Mux_FMConf_t;
```

```
/*
** SONET/SDH Adaptation Interface Fault Management Configuration
*/



/*
** Configuration SONET/SDH Server Signal Failure reporting
*/

typedef enum {
NPF_IF_SONET_SDH_ADAPT_SSF_ReportDisable = 0,
/* Interface configured not to report
** Server Signal Failure
*/
NPF_IF_SONET_SDH_ADAPT_SSF_ReportEnable = 1
/* Interface configured to report SSF */
} NPF_IfSONET_SDH_Adapt_SSF_RepMode_t;


/*
**  Configuration of SONET/SDH Alarm Indication Signal (AIS) reporting
Mode
*/

typedef enum {
NPF_IF_SONET_SDH_ADAPT_AIS_ReportDisable = 0,
/* Interface configured not to report AIS signal */
NPF_IF_SONET_SDH_ADAPT_AIS_ReportEnable = 1
/* Interface configured to report AIS signal */
} NPF_IfSONET_SDH_Adapt_AIS_RepMode_t;

/*
** Configuration SONET/SDH Remote Defect Information reporting
*/
typedef enum {
NPF_IF_SONET_SDH_ADAPT_RDI_ReportDisable = 0,
/* Inteface configured not to report Remote Defect Information */
NPF_IF_SONET_SDH_ADAPT_RDI_ReportEnable = 1
/* Interface configured to report Remote Defect Information */
} NPF_IfSONET_SDH_Adapt_RDI_RepMode_t;


/*
**  Configuration of SONET/SDH Alarm Indication Signal (AIS) reporting
Mode
*/

typedef struct {

 NPF_IfSONET_SDH_Adapt_AIS_RepMode_t         AIS_AdaptReportMode;
 /* Alarm Indication Signal */
 NPF_IfSONET_SDH_Adapt_RDI_RepMode_t         RDI_AdaptReportMode;
 /* Remote Defect Information */
 NPF_IfSONET_SDH_Adapt_SSF_RepMode_t         SSF_AdaptReportMode;
 /* Server Signal Fail */
```

```
 }NPF_IfSONET_SDH_Adapt_FMConf_t;



/*
** SONET-SDH Mapping (Path) Fault Management Structures
*/



/*
** Configuration SONET/SDH Server Signal Failure reporting
*/

typedef enum {
NPF_IF_SONET_SDH_MAP_SSF_ReportDisable = 0,
 /* Interface configured not to report
 ** Server Signal Failure
 */
NPF_IF_SONET_SDH_MAP_SSF_ReportEnable = 1
 /* Interface configured to report SSF */
} NPF_IfSONET_SDH_Map_SSF_RepMode_t;
/*
**  Configuration of SONET/SDH Alarm Indication Signal (AIS) reporting
Mode
*/

typedef enum {
NPF_IF_SONET_SDH_MAP_AIS_ReportDisable = 0,
/* Interface configured not to report AIS signal */
NPF_IF_SONET_SDH_MAP_AIS_ReportEnable = 1
/* Interface configured to report AIS signal */
} NPF_IfSONET_SDH_Map_AIS_RepMode_t;

/*
** Configuration SONET/SDH Remote Defect Information reporting
*/
typedef enum {
NPF_IF_SONET_SDH_MAP_RDI_ReportDisable = 0,
/* Inteface configured not to report Remote Defect Information */
NPF_IF_SONET_SDH_MAP_RDI_ReportEnable = 1

/* Interface configured to report Remote Defect Information */
} NPF_IfSONET_SDH_Map_RDI_RepMode_t;

typedef enum {
NPF_IF_SONET_SDH_MAP_TIMdis = 0,  /* TIM processing disabled */
NPF_IF_SONET_SDH_MAP_TIMen = 1    /*  TIM processing enabled */
} NPF_IfSONET_SDH_Map_TIMCfgSet_t ;

typedef enum {
NPF_IF_SONET_SDH_MAP_TTI_AISdis = 0,
/* AIS insertion on HO Path, TTI mismatch disabled */
NPF_IF_SONET_SDH_MAP_TTI_AISen = 1
```

```
/* AIS insertion on Ho Path TTI mismatch enabled */
} NPF_IfSONET_SDH_Map_TTI_AISCfgSet_t ;


/*
**  SONET-SDH Mapping Interface Signal Degeneration Threshold
*/
typedef enum {

  NPF_IF_SONET_SDH_MAP_ThreshBursty = 5,

  NPF_IF_SONET_SDH_MAP_ThreshPoisson = 6

} NPF_IfSONET_SDH_MapThresh_t;

/*
** Multiplexing Interface FM structure
*/

typedef struct {

 NPF_IfSONET_SDH_Map_AIS_RepMode_t         AIS_MapReportMode;
  /* Alarm Indication Signal */
 NPF_IfSONET_SDH_Map_RDI_RepMode_t         RDI_MapReportMode;
  /* Remote Defect Information */
 NPF_IfSONET_SDH_Map_SSF_RepMode_t         SSF_MapReportMode;
  /* Server Signal Fail */
 NPF_IfSONET_SDH_Map_TIMCfgSet_t         Map_TIMCfgSet;

 NPF_IfSONET_SDH_Map_TTI_AISCfgSet_t  Map_TTIAISCfgSet;

 NPF_IfSONET_SDH_MapThresh_t     MapThreshold;
                        /* Mapping Error Threshold */

}NPF_IfSONET_SDH_Map_FMConf_t;


/*
** SONET/SDH Medium Interface Status
*/

typedef enum {

NPF_IF_SONET_SDH_Medium_NoDefect = 1,   /* No Defect */
NPF_IF_SONET_SDH_Medium_LOS = 2,    /* Loss of Signal Defect */
NPF_IF_SONET_SDH_Medium_LOF = 3   /* Loss of Frame Defect */

} NPF_IfSONET_SDH_MediumStatusBits_t;

struct NPF_IfSONET_SDH_MediumStatus{
    NPF_uint16_t               n_data;
    NPF_IfSONET_SDH_MediumStatusBits_t   *statusinfo;
    } ;


/*
```

```
** SONET/SDH Medium Current Status
*/

struct NPF_IfSONET_SDH_MediumCurrStatus{

    NPF_uint32_t   MediumTimeElapsed;         /* Time Elapsed */
    NPF_uint32_t   MediumValidIntervals;      /* Valid Intervals */
    NPF_uint32_t   MediumInvalidIntervals;    /* Invalid Intervals */

} ;



/*
** SONET/SDH Section Interface Status
*/


typedef enum {

NPF_IF_SONET_SDH_Section_NoDefect = 1,  /* Section No Defect */
NPF_IF_SONET_SDH_Section_Status_TIM = 2,   /* Trace Identifier Mismatch
Alarm  */
NPF_IF_SONET_SDH_Section_Status_SSF    = 3 /* Server Signal Failure */

} NPF_IfSONET_SDH_SectionStatusType_t;

struct NPF_IfSONET_SDH_SectionStatus{
    NPF_uint16_t               n_data;
    NPF_IfSONET_SDH_SectionStatusType_t   *statusinfo;
    } ;

/*
**  SONET-SDH Multiplexing Interface Status
*/


typedef enum {

NPF_IF_SONET_SDH_Mux_NoDefect = 1,  /* No Defect */
NPF_IF_SONET_SDH_Mux_AIS = 2,           /* Mux (Line) AIS */
NPF_IF_SONET_SDH_Mux_RDI = 3,       /* Mux (Line) RDI */
NPF_IF_SONET_SDH_Mux_SSF = 4,       /* Multiplexing Server Signal Fail
*/
/*
*/
NPF_IF_SONET_SDH_Mux_EXC = 5,       /* Mux Excessive Bit Error Rate */
NPF_IF_SONET_SDH_Mux_DEG = 6        /* Multiplexing Signal Degrade */

} NPF_IfSONET_SDH_MuxStatusType_t;

struct NPF_IfSONET_SDH_MuxStatus{
    NPF_uint16_t               n_data;
    NPF_IfSONET_SDH_MuxStatusType_t   *statusinfo;
    } ;
```

```
/*
** SONET/SDH Adaptation Interface Status
*/

typedef enum {

NPF_IF_SONET_SDH_Adapt_Status = 0,

NPF_IF_SONET_SDH_Adapt_NoDefect = 1,       /* No Defect */
NPF_IF_SONET_SDH_Adapt_Status_AIS = 2,     /* Adaptation (line) AIS */
NPF_IF_SONET_SDH_Adapt_Status_RDI = 3,     /* Adaptation RDI */
NPF_IF_SONET_SDH_Adapt_Status_SSF = 4 ,    /* Adaptation Server Signal
Fail */
NPF_IF_SONET_SDH_Adapt_Status_LOP = 5      /* Adaptation Loss of
Pointer */

} NPF_IfSONET_SDH_AdaptStatusType_t;

struct NPF_IfSONET_SDH_AdaptStatus{
    NPF_uint16_t               n_data;
    NPF_IfSONET_SDH_AdaptStatusType_t   *statusinfo;
    } ;



/*
** SONET/SDH Mapping Interface Status
*/

typedef enum {

NPF_IF_SONET_SDH_Map_NoDefect   = 1,    /* No Defect */
NPF_IF_SONET_SDH_Status_Map_AIS = 2,       /* Path AIS */
NPF_IF_SONET_SDH_Status_Map_RDI = 3,       /* Path RDI */
NPF_IF_SONET_SDH_Status_Map_SSF = 4,       /* Path Server Signal Fail
*/
/*
*/
NPF_IF_SONET_SDH_Status_Map_EXC = 5,   /* Excessive Bit Error Rate */
NPF_IF_SONET_SDH_Status_Map_DEG = 6,       /* Signal Degrade */
/*
*/
NPF_IF_SONET_SDH_Status_Map_RFI = 7,   /* Path RFI */
/*
*/
NPF_IF_SONET_SDH_Status_Map_UNEQ = 8,  /* Map Status Unequipped */
NPF_IF_SONET_SDH_Status_Map_TIM = 9,   /* TTI Mismatch defect */
NPF_IF_SONET_SDH_Status_Map_PLM = 10,  /* Payload Mismatch
Defect(Corresp. to SLM )*/
NPF_IF_SONET_SDH_Status_Map_LOM = 11   /* Loss of Multi-frame */

} NPF_IfSONET_SDH_MapStatusType_t;

struct NPF_IfSONET_SDH_MapStatus{
```

```
    NPF_uint16_t               n_data;
    NPF_IfSONET_SDH_MapStatusType_t   *statusinfo;
    };


/*
** --- End of SONET/SDH Fault Management Data Structures
*/



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

#define NPF_IF_TYPE_SONET_SDH  10         /* SONET-SDH Interface */

/*
** End of Interface types for SONET/SDH
*/

typedef enum {
/*
** SONET-SDH Interface types
*/
NPF_IF_TYPE_SONET_SDH_Medium=1,     /* SONET-SDHPhysical Medium  */
NPF_IF_TYPE_SONET_SDH_Section=2,    /* SONET-SDHSection Interface */
NPF_IF_TYPE_SONET_SDH_Mux =3,       /* SONET-SDHMultiplexing interface
*/
NPF_IF_TYPE_SONET_SDH_Adaptation =4,    /* SONET-SDHAdaptation
interface */
NPF_IF_TYPE_SONET_SDH_Mapping=5 /* SONET-SDHMapping interface */
/*
** End of SONET-SDH Interface types
*/
} NPF_IfSONET_SDH_Type_t;



/*
**  SONET-SDH Medium Encoding
*/
typedef enum {
  NPF_IF_SONET_SDH_MediumCoding_Other = 1,  /* Coding is other */
  NPF_IF_SONET_SDH_MediumCoding_B3ZS = 2,   /* Coding is B3ZS */
  NPF_IF_SONET_SDH_MediumCoding_CMI = 3,    /* Coding is CMI */
  NPF_IF_SONET_SDH_MediumCoding_NRZ = 4,    /* Coding is NRZ */
  NPF_IF_SONET_SDH_MediumCoding_RZ = 5      /* Coding is RZ */
} NPF_IfSONET_SDH_MediumCoding_t;


/*
**  SONET-SDH Medium Line Type
*/
typedef enum {
  NPF_IF_SONET_SDH_MediumLine_Other = 1,
```

```
/* Medium Line type is Other */
  NPF_IF_SONET_SDH_MediumLine_ShortSingleMode = 2,
/* Medium Line type is Short Single Mode */
  NPF_IF_SONET_SDH_MediumLine_LongSingleMode = 3,
/* Medium Line type is Long Single Mode */
  NPF_IF_SONET_SDH_MediumLine_MultiMode = 4,
/* Medium Line type is Multi Mode */

  NPF_IF_SONET_SDH_MediumLine_Coax = 5,
/* Medium Line type is Coax */
  NPF_IF_SONET_SDH_MediumLine_UTP = 6
/* Medium Line type is UTP */
} NPF_IfSONET_SDH_MediumLineType_t;


/*
**  SONET-SDH Medium Loopback Configuraton
*/
typedef enum {
  NPF_IF_SONET_SDH_NoLoop = 0,
/* NO Loop */
  NPF_IF_SONET_SDH_FacilityLoop = 1,
/* SONET Facility Loop */
  NPF_IF_SONET_SDH_TerminalLoop = 2,
/* SONET Terminal Loop */
  NPF_IF_SONET_SDH_OtherLoop = 3
/* SONET Other Loop */
} NPF_IfSONET_SDH_MediumLoopbackConfig_t;


/*
** SONET/SDH Medium Severely Errored Seconds (SES) Threshold
*/
typedef enum {
  NPF_IF_SONET_SDH_ThreshOther = 0,
/* Other */
  NPF_IF_SONET_SDH_ThreshBellcore1991 = 1,
/* Bellcore TR-NWT-000253, 1991... */
  NPF_IF_SONET_SDH_ThreshAnsi1993 = 2,
/* ANSI T1.231, 1993, ...  */
  NPF_IF_SONET_SDH_ThreshItu1995 = 3,
/* ITU-T G.826, 1995 ... */
  NPF_IF_SONET_SDH_ThreshAnsi1997 = 4,
/* ANSI T1.231, 1997 ... */
  NPF_IF_SONET_SDH_ThreshBursty = 5,

  NPF_IF_SONET_SDH_ThreshPoisson = 6

} NPF_IfSONET_SDH_MediumSESthresholdSet_t;



/*
** SONET-SDH  Physical Medium Interface Attributes
*/
typedef struct {
```

```
         NPF_uint32_t               port;             /* Physical Port
Number */
         NPF_IfSONET_SDH_MediumCoding_t  MediumLineCoding;  /* Medium
Line Coding */
         NPF_IfSONET_SDH_MediumLineType_t MediumLineType;   /* Medium
Line Type */
         NPF_IfSONET_SDH_MediumLoopbackConfig_t MediumLoopbackConfig ;
                                          /* Medium Loop-back
Configuration */
         NPF_IfSONET_SDH_MediumSESthresholdSet_t MediumSESthresholdSet
;
                              /* Medium SES recognized set of
thresholds */
           NPF_IfSONET_SDH_MediumStatus_t  MediumStatus;
                                                /* Medium Status */
         NPF_IfSONET_SDH_MediumCurrStatus_t  MediumCurrentStatus;
                                          /* Medium Current Status */

         } NPF_IfSONET_SDH_Medium_t;


/*
**  SONET Section or SDH Regenerator Section Interface
*/


/*
**  SONET-SDH Section Type
*/
typedef enum {
  NPF_IF_SONETSDH_Section_SONET = 1,  /* Section is SONET */
  NPF_IF_SONETSDH_Section_SDH = 2     /* Section is SDH */
 } NPF_IfSONET_SDH_SectionType_t;

/*
** SDH Regenerator Section
*/
typedef enum {

  NPF_IF_SDH_RegSection_none = 0,   /* Type not defined */
  NPF_IF_SDH_RegSection_STM_0 = 1,   /* STM-0 51.84Mbps */
  NPF_IF_SDH_RegSection_STM_1 = 2,  /* STM-1 155.55Mbps */
  NPF_IF_SDH_RegSection_STM_4 = 3,  /* STM-4 622.08Mbps */
  NPF_IF_SDH_RegSection_STM_16 = 4, /* STM-16 2.48Gbps */
  NPF_IF_SDH_RegSection_STM_64 = 5, /* STM-64 9.92Gbps */
  NPF_IF_SDH_RegSection_STM_256 = 6 /* STM-256 39.68Gbps */


} NPF_IfSDH_RegSectionIf_t;

/*
** SONET Section
*/
typedef enum {

  NPF_IF_SONET_Section_none = 0,    /* Type not defined */
```

```
  NPF_IF_SONET_Section_STS_1 = 1,    /* STS1 Section 51.84Mbps */
  NPF_IF_SONET_Section_STS_3 =  2,  /* STS3 Section 155.55Mbps */
  NPF_IF_SONET_Section_STS_12 = 3,  /* STS12 Section 622.08Mbps */
  NPF_IF_SONET_Section_STS_48 = 4,  /* STS48 Section 2.48Gbps */
  NPF_IF_SONET_Section_STS_192 = 5, /* STS192 Section 9.92Gbps */
  NPF_IF_SONET_Section_STS_768 = 6  /* STS768 Section 39.68Gbps */

} NPF_IfSONET_SectionIf_t;


/*
**  SONET Section or SDH Regenerator Section Interface Attributes
*/
typedef struct {
   NPF_IfSONET_SDH_SectionType_t    SectionType;
 union{
  NPF_IfSONET_SectionIf_t   SectionIf;
  NPF_IfSDH_RegSectionIf_t  RegSectionIf;
 }u;
 NPF_IfSONET_SDH_Section_FMConf_t    Section_FM_Conf;
 NPF_IfSONET_SDH_SectionStatus_t  SectionCurStatus;
}NPF_IfSONET_SDH_Section_t;


/*
** SDH High Order Multiplexing Interface Attributes
*/
typedef enum {

  NPF_IF_SDH_Mux_none = 0,  /* Type of multiplexing not defined */
  NPF_IF_SDH_Mux_AUG_1 = 1, /* Multiplexing in AUG 1 */
  NPF_IF_SDH_Mux_AUG_4 = 2, /* Multiplexing in AUG 4 */
  NPF_IF_SDH_Mux_AUG_16 = 3,    /* Multiplexing in AUG 16 */
  NPF_IF_SDH_Mux_Aug_64 = 4,    /* Multiplexing in AUG 64 */
  NPF_IF_SDH_Mux_Aug_256 = 5    /* Multiplexing in AUG 256 */

}NPF_IfSDH_HO_MuxIf_t;

/*
** SONET STS Group Multiplexing Interface Attributes
*/
typedef enum {
  NPF_IF_SONET_Mux_none = 0,    /* Type of multiplexing not defined */
  NPF_IF_SONET_Mux_STS_1_Group = 1, /* Multiplexing into STS-1  */
  NPF_IF_SONET_Mux_STS_3_Group = 2, /* Multiplexing into STS-3  */
  NPF_IF_SONET_Mux_STS_12_Group = 3,    /* Multiplexing into STS-12 */
  NPF_IF_SONET_Mux_STS_48_Group = 4,    /* Multiplexing into STS-48 */
  NPF_IF_SONET_Mux_STS_192_Group = 5, /* Multiplexing into STS-192 */
  NPF_IF_SONET_Mux_STS_768_Group = 6  /* Multiplexing into STS-768 */

}NPF_IfSONET_STS_MuxIf_t;


/*
** SDH Low Order Multiplexing Interface Attributes
*/
```

```
typedef enum {

  NPF_IF_SDH_Mux_TUG_2 = 6, /* Multiplexing in TUG 2 */
  NPF_IF_SDH_Mux_TUG_3 = 7  /* Multiplexing in TUG 3 */

} NPF_IfSDH_LO_MuxIf_t;

/*
** SONET VT Group Multiplexing Interface Attributes
*/
typedef enum {

  NPF_IF_SONET_Mux_VT_Group = 7    /* Multiplexing into VT Group */

}NPF_IfSONET_VT_MuxIf_t;



typedef enum{

      NPF_IF_SONET_SDH_WORKING = 1, /* Working Trail of a APS
combination */
      NPF_IF_SONET_SDH_PROTECTING = 2, /* The Protecting Trail of a APS
combination */
      NPF_IF_SONET_SDH_NORMAL = 3, /* This Multiplex Section is
protected */
      NPF_IF_SONET_SDH_EXTRA_TRAFFIC=4 /* This Multiplex Section
carries extra-traffic */

} NPF_IfSONET_SDH_Protection_Flags_t;

typedef enum {

      NPF_IF_SONET_SDH_TRAIL_PROT = 1,/* Trail Protection */
      NPF_IF_SONET_SDH_SUBNETWORK = 2 /* Subnet Protection */

} NPF_IfSONET_SDH_Protection_Type_t;

typedef enum {

      NPF_IF_SONET_SDH_ONE_ONE = 1, /* 1+1 Protection */
      NPF_IF_SONET_SDH_ONE_N = 2      /* 1:N Protection */


} NPF_IfSONET_SDH_Protection_Arch_t;

typedef enum {

      NPF_IF_SONET_SDH_UNI_D = 1,   /* Unidirectional */
      NPF_IF_SONET_SDH_BI_D = 2     /* Bidirectional */

} NPF_IfSONET_SDH_Protection_Switch_t;

typedef enum {

      NPF_IF_SONET_SDH_REVERTIVE = 1,           /*  */
```

```
      NPF_IF_SONET_SDH_NON_REVERTIVE = 2  /*  */

} NPF_IfSONET_SDH_Protection_Operation_t;



/*
** SONET-SDH Interface Protection Information
*/

typedef struct {
           NPF_IfSONET_SDH_Protection_Flags_t   ProtFlags;
           NPF_IfSONET_SDH_Protection_Type_t    ProtType;
           NPF_IfSONET_SDH_Protection_Arch_t    ArchType;
           NPF_IfSONET_SDH_Protection_Switch_t  SwitchType;
           NPF_IfSONET_SDH_Protection_Operation_t  OperType;
           NPF_boolean_t NPF_IfSONET_SDH_Protection_Signal_t;
           } NPF_IfSONET_SDH_Protection_t;



/*
**  SONET-SDH Multiplexing Interface attributes,
*/
typedef struct {

union {

 NPF_IfSDH_HO_MuxIf_t         SDH_HO_Multiplexing;
 NPF_IfSDH_LO_MuxIf_t         SDH_LO_Multiplexing;
 NPF_IfSONET_STS_MuxIf_t      SONET_STS_Multiplexing;
 NPF_IfSONET_VT_MuxIf_t SONET_VT_Multiplexing;

 } umux;

           NPF_IfSONET_SDH_Protection_t   Mux_Protection;
           NPF_IfSONET_SDH_Mux_FMConf_t         Mux_FM_Conf;
           NPF_IfSONET_SDH_MuxStatus_t          MultiplexingStatus;

 } NPF_IfSONET_SDH_Multiplexing_t;

/*
** SDH High Order Adaptation Attributes
*/
typedef enum {

  NPF_IF_SDH_Adapt_none = 0,    /* Type of adaptation not defined */
  NPF_IF_SDH_Adapt_AU_3 = 1,        /* AU-3  51.84Mbps */
  NPF_IF_SDH_Adapt_AU_4 = 2,        /* AU-4  155.55Mbps */
  NPF_IF_SDH_Adapt_AU_4_4c = 3, /* AU-4-4c 311.04Mbps */
  NPF_IF_SDH_Adapt_AU_4_16c = 4,    /* AU-4-16c 622.08Mbps */
  NPF_IF_SDH_Adapt_AU_4_64c = 5,    /* AU-4-64c 2.48Gbps */
  NPF_IF_SDH_Adapt_AU_4_256c = 6    /* AU-4-256c 39.68Gbps */

} NPF_IfSDH_HO_AdaptIf_t;
```

```
/*
** SONET High Order Adaptation Attributes
*/
typedef enum {

  NPF_IF_SONET_Adapt_none = 0,  /* Type of adaptation not defined */

  NPF_IF_SONET_Adapt_STS_1 = 1,
/* Adapted SPE is STS1 51.84Mbps */
  NPF_IF_SONET_Adapt_STS_3c = 2,
/* Adapted SPE is STS3c 155.55Mbps */

  NPF_IF_SONET_Adapt_STS_12c = 3,
/* Adapted SPE is STS12c 622.08Mbps */
  NPF_IF_SONET_Adapt_STS_48c = 4,
/* Adapted SPE is STS48c 2.48Gbps */
  NPF_IF_SONET_Adapt_STS_192c = 5,
/* Adapted SPE is STS192c 9.92Gbps */
  NPF_IF_SONET_Adapt_STS_768c = 6
/* Adapted SPE is STS768c 39.68Gbps */

} NPF_IfSONET_STS_AdaptIf_t;


/*
** SDH Low Order Adaptation Interface Attributes
*/
typedef enum {

  NPF_IF_SDH_Adapt_TU_11 = 7,   /* Adapt Low Order VC-11 - TU-11 */
  NPF_IF_SDH_Adapt_TU_12 = 8,   /* Adapt Low Order VC-12 -  TU-12 */
  NPF_IF_SDH_Adapt_TU_2 = 9,    /* Adapt Low Order VC-2 - TU-2 */
  NPF_IF_SDH_Adapt_TU_3 = 10    /* Adapt Low Order VC-3 - TU-3 */

} NPF_IfSDH_LO_AdaptIf_t;


/*
** SONET VT Adaptation Interface
*/
typedef enum {

  NPF_IF_SONET_VT15 = 7,    /* SONET VT1.5 */
  NPF_IF_SONET_VT2 = 8, /* SONET VT2 */
  NPF_IF_SONET_VT3 = 9, /* SONET VC3 */
  NPF_IF_SONET_VT6 = 10    /* SONET VT6 */

} NPF_IfSONET_VT_AdaptIf_t;



/*
**  SONET-SDH Adaption Interface attributes,
**
*/
```

```
typedef struct {

 union {

 NPF_IfSDH_HO_AdaptIf_t SDH_HO_Adaptation;
 NPF_IfSDH_LO_AdaptIf_t SDH_LO_Adaptation;
 NPF_IfSONET_STS_AdaptIf_t  SONET_STS_Adaptation;
 NPF_IfSONET_VT_AdaptIf_t   SONET_VT_Adaptation;

 } u;
    NPF_IfSONET_SDH_Adapt_FMConf_t    Adapt_FM_Conf;
    NPF_IfSONET_SDH_AdaptStatus_t    AdaptationStatus;

                 } NPF_IfSONET_SDH_Adaptation_t;


/*
** SDH High Order Mapping Attributes
*/


typedef enum {


  NPF_IF_SDH_Map_VC_3 = 1,       /* VC-3  51.84Mbps */
  NPF_IF_SDH_Map_VC_4 = 2,       /* VC-4  155.55Mbps */
  NPF_IF_SDH_Map_VC_4_4c = 3,    /* VC-4-4c 311.04Mbps */
  NPF_IF_SDH_Map_VC_4_16c = 4,  /* VC-4-16c 622.08Mbps */
  NPF_IF_SDH_Map_VC_4_64c = 5,  /* VC-4-64c 2.48Gbps */
  NPF_IF_SDH_Map_VC_4_256c = 6,  /* VC-4-256c 39.68Gbps */

  NPF_IF_SDH_Map_VCG = 255    /* VCG, bandwidth provided by the summ of
the */
           /* bandwidth of all parent Mapping interfaces*/

} NPF_IfSDH_HO_MapIf_t;


/*
** SONET STS Mapping Attributes
*/



typedef enum {

  NPF_IF_SONET_Map_STS_1_SPE = 1,     /* SPE is STS1 51.84Mbps */
  NPF_IF_SONET_Map_STS_3c_SPE = 2,  /* SPE is STS3c 155.55Mbps */
  NPF_IF_SONET_Map_STS_12c_SPE = 3, /* SPE is STS12c 622.08Mbps */
  NPF_IF_SONET_Map_STS_48c_SPE = 4, /* SPE is STS48c 2.48Gbps */
  NPF_IF_SONET_Map_STS_192c_SPE = 5,     /* SPE is STS192c 9.92Gbps */
  NPF_IF_SONET_Map_STS_768c_SPE = 6, /* SPE is STS768c 39.68Gbps */
  NPF_IF_SONET_Map_VCG = 255         /* VCG, bandwidth provided by the
summ of the */
                                     /* bandwidth of all parent Mapping
interfaces */
```

```
} NPF_IfSONET_STS_MapIf_t;



/*
** SDH Low Order Mapping Interface Attributes
*/
typedef enum {


  NPF_IF_SDH_Map_VC11 = 7,    /* Map Low Order VC11 */
  NPF_IF_SDH_Map_VC12 = 8,    /* Map Low Order VC12 */
  NPF_IF_SDH_Map_VC2 = 9,     /* Map Low Order VC2 */

  NPF_IF_SDH_Map_LO_VCG = 255 /* VCG, bandwidth provided by the summ of
the */
                           /* bandwidth of all parent Mapping
interfaces */
} NPF_IfSDH_LO_MapIf_t;


/*
** SONET VT Mapping Interface
*/

typedef enum {

  NPF_IF_SONET_VT15_SPE = 7,        /* SONET VC11 */
  NPF_IF_SONET_VT2_SPE = 8,         /* SONET VC12 */
  NPF_IF_SONET_VT3_SPE = 9,         /* SONET VC3 */
  NPF_IF_SONET_VT_6_SPE = 10,       /* SONET VT6 */

  NPF_IF_SONET_Map_LO_VCG = 255     /* VCG, bandwidth provided by the
sum of the */
           /* bandwidth of all parent Mapping interfaces */
} NPF_IfSONET_VT_MapIf_t;



/*
** VCG Attributes
*/
typedef struct {

 NPF_boolean_t  NPF_IF_SONET_SDH_TSD_ENABLE;   /* TSD enable */
 NPF_boolean_t  NPF_IF_SONET_SDH_LCAS_ENABLE;  /* LCAS Enable*/

} NPF_IfSONET_SDH_VCG_Attrib_t;



/*
**      SONET-SDH  Mapping Interface attributes,
**
```

```
*/
typedef struct {
 union {
 NPF_IfSDH_HO_MapIf_t         SDH_HO_Mapping;   /* SDH High Order */
 NPF_IfSDH_LO_MapIf_t         SDH_LO_Mapping;         /* SDH Low Order
*/
 NPF_IfSONET_STS_MapIf_t      SONET_STS_Mapping;      /* SONET STS */
 NPF_IfSONET_VT_MapIf_t       SONET_VT_Mapping; /* SONET VT */
       } u;
           NPF_IfSONET_SDH_Protection_t   Map_Protection;
           NPF_IfSONET_SDH_VCG_Attrib_t   VCGAttrib;
           NPF_IfSONET_SDH_MapStatus_t   MapStatus;
           /* Path Current Status */

                 } NPF_IfSONET_SDH_Mapping_t;


/*
**  +++ SONET-SDH Interface Attributes
*/
 struct NPF_IfSONET_SDH {

    NPF_IfSONET_SDH_Type_t    SONET_SDH_IfType; /* SONET-SDH Interface
type */

        union {

           NPF_IfSONET_SDH_Medium_t SONET_SDH_Medium;

/* SONET-SDHPhysical Medium Interface Attributes */

           NPF_IfSONET_SDH_Section_t SONET_SDH_Section;

/* SONET Section SDH Regenerator Section Interface Attributes */

           NPF_IfSONET_SDH_Multiplexing_t SONET_SDH_Multiplexing;

/* SONET-SDHMultiplexing Attributes */

           NPF_IfSONET_SDH_Adaptation_t SONET_SDH_Adaptation;


/* SONET-SDHAdaptation Attributes */

           NPF_IfSONET_SDH_Mapping_t SONET_SDH_Mapping;

/* SONET-SDHMapping Attributes */
   }u;
 };


/*
** --- End of SONET_SDH Interface Attributes Definitions
*/
```

```
/*
** --- SONET-SDH Performance Monitoring Data Structures
*/
/*
** +++ SONET-SDH Performance Monitoring Configuration Data Structures
*/


/*
** +++ SONET-SDH Performance Monitoring Statistics Data Structures
*/

/*
 *  SONET-SDH Section Statistics
 */

typedef struct {

NPF_uint64_t    SectionCurrentESs;
/* section current Error Seconds */
NPF_uint64_t    SectionCurrentSESs;
/* section current Severely Errored Seconds */
NPF_uint64_t    SectionCurrentSEFSs;
/* section Severely Errored Framing Seconds */
NPF_uint64_t    SectionCurrentCVs;
/* section current Coding Violation */
} NPF_IfSONET_SDH_SectionStats_t;

/*
 *  SONET-SDH Interval Section Statistics
 */

typedef struct {

 NPF_uint32_t    SectionIntervalID;
/* section Interval Number (0-96)*/
 NPF_boolean_t   SectionIntervalValidData;
/* section Interval Valid Data */

 NPF_IfSONET_SDH_SectionStats_t IntervalSectionStats;
} NPF_IfSONET_SDH_SectionIntervalStats_t;

/*
** SONET-SDH Section Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_SectionStatistics{

      NPF_IfSONET_SDH_SectionStats_t
      CurrentSectionStatistics;
      NPF_IfSONET_SDH_SectionIntervalStats_t
      IntervalSectionStatistics;

      } ;
```

```
/*
 *  SONET-SDH Mux Statistics
 */
typedef struct {
                    NPF_uint64_t    MuxCurrentESs;
 /* Mux current Error Seconds */
             NPF_uint64_t    MuxCurrentSESs;
/* Mux current Severely Errored Seconds */
             NPF_uint64_t    MuxCurrentCVs;
/* Mux current Coding Violation */
             NPF_uint64_t    MuxCurrentUAs;
/* Mux current Unavailable Seconds */

} NPF_IfSONET_SDH_MuxBasicStats_t;

typedef struct {
   NPF_uint32_t    MuxIntervalID;
                     /* Mux Interval Number (0-96)*/
   NPF_boolean_t   MuxIntervalValidData;
                     /* Mux Interval Valid Data */
   NPF_IfSONET_SDH_MuxBasicStats_t   MuxIntervalStats;
  } NPF_IfSONET_SDH_MuxIntervalStats_t;

typedef struct {

 NPF_IfSONET_SDH_MuxBasicStats_t    MuxCurrentStats;
 NPF_IfSONET_SDH_MuxIntervalStats_t  MuxIntervalStats;

} NPF_IfSONET_SDH_MuxStats_t;


/*
** SONET-SDH Multiplexing Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_MuxStatistics{

    NPF_IfSONET_SDH_MuxStats_t  LocalMuxStatistics;
    NPF_IfSONET_SDH_MuxStats_t  FarEndMuxStatistics;

    } ;


/*
 *  SONET-SDH Path Statistics
 */
typedef struct {

NPF_uint64_t    PathCurrentESs;
/* Path current Error Seconds */
NPF_uint64_t    PathCurrentSESs;
/* Path current Severely Errored Seconds */
NPF_uint64_t    PathCurrentCVs;
/* Path current Coding Violation */
```

```
NPF_uint64_t    PathCurrentUAs;
/* Path current Unavailable Seconds */

} NPF_IfSONET_SDH_PathBasicStats_t;

typedef struct {

NPF_uint32_t    PathIntervalID;
/* Path Interval Number (0-96)*/
NPF_boolean_t  PathIntervalValidData;
/* Path Interval Valid Data */

 NPF_IfSONET_SDH_PathBasicStats_t  MapIntervalStats;

} NPF_IfSONET_SDH_PathIntervalStats_t;

typedef struct {

 NPF_IfSONET_SDH_PathBasicStats_t PathCurrentStats;
 NPF_IfSONET_SDH_PathIntervalStats_t PathIntervalStats;

} NPF_IfSONET_SDH_PathStats_t;


/*
** SONET-SDH Mapping Interface Performance Monitoring Statistics
*/
struct NPF_IfSONET_SDH_PathStatistics{
      NPF_IfSONET_SDH_PathStats_t  LocalPathStatistics;
      NPF_IfSONET_SDH_PathStats_t  FarEndPathStatistics;
      };

/*
** --- End of SONET-SDH Performance Monitoring Data Structures
*/

/*
** --- Completion Callbacks Data Structures
*/

/*
 *  Completion Callback Types

 */
/*
 *    Completion Callback Types for SONET-SDH
*/
#define NPF_IF_SONET_SDH_MediumTypeAttrSet
((NPF_IF_TYPE_SONET_SDH<<16)+1)
#define NPF_IF_SONET_SDH_MediumCodingAttrSet
((NPF_IF_TYPE_SONET_SDH<<16)+2)
#define NPF_IF_SONET_SDH_MediumLineTypeAttrSet
(NPF_IF_TYPE_SONET_SDH<<16)+3)
#define NPF_IF_SONET_SDH_MediumLoopbackConfigAttrSet
((NPF_IF_TYPE_SONET_SDH<<16)+4)
#define NPF_IF_SONET_SDH_MediumSESthresAttrSet
```

```
((NPF_IF_TYPE_SONET_SDH<<16)+5)
#define NPF_IF_SONET_SDH_MediumStatusGet
((NPF_IF_TYPE_SONET_SDH<<16)+6)
#define NPF_IF_SONET_SDH_MediumCurrStatusGet
((NPF_IF_TYPE_SONET_SDH<<16)+7)

/**/
#define NPF_IF_SONET_SDH_SectionAttrSet ((NPF_IF_TYPE_SONET_SDH<<16)+8)
#define NPF_IF_SONET_SDH_SectionFMConfigSet
((NPF_IF_TYPE_SONET_SDH<<16)+9)
#define NPF_IF_SONET_SDH_SectionStatusGet
((NPF_IF_TYPE_SONET_SDH<<16)+10)
#define NPF_IF_SONET_SDH_SectionStatisticsQuery
((NPF_IF_TYPE_SONET_SDH<<16)+11)

/**/
#define NPF_IF_SONET_SDH_MuxAttrSet ((NPF_IF_TYPE_SONET_SDH<<16)+12)
#define NPF_IF_SONET_SDH_MuxFMConfigSet
((NPF_IF_TYPE_SONET_SDH<<16)+13)
#define NPF_IF_SONET_SDH_MuxStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+14)
#define NPF_IF_SONET_SDH_MuxStatisticsQuery
((NPF_IF_TYPE_SONET_SDH<<16)+15)
/**/
#define NPF_IF_SONET_SDH_AdaptAttrSet ((NPF_IF_TYPE_SONET_SDH<<16)+16)
#define NPF_IF_SONET_SDH_AdaptFMConfigSet
((NPF_IF_TYPE_SONET_SDH<<16)+17)
#define NPF_IF_SONET_SDH_AdaptStatusGet
((NPF_IF_TYPE_SONET_SDH<<16)+18)
/**/
#define NPF_IF_SONET_SDH_MapAttrSet ((NPF_IF_TYPE_SONET_SDH<<16)+19)

#define NPF_IF_SONET_SDH_MapFMConfigSet
((NPF_IF_TYPE_SONET_SDH<<16)+20)
#define NPF_IF_SONET_SDH_MapStatusGet ((NPF_IF_TYPE_SONET_SDH<<16)+21)

#define NPF_IF_SONET_SDH_MapStatisticsQuery
((NPF_IF_TYPE_SONET_SDH<<16)+22)



/*
** End of SONET-SDH Completion Callback types
*/
#ifdef temporary_no_core
/*
** Asynchronous Response types for SONET-SDH interfaces
*/
         /* Fault Management Status */
         NPF_IfSONET_SDH_MediumStatus_t         *if_MediumStatus;
         NPF_IfSONET_SDH_MediumCurrStatus_t     *if_MediumCurrStatus;
         NPF_IfSONET_SDH_SectionStatus_t        *if_SectionStatus;
         NPF_IfSONET_SDH_MuxStatus_t            *if_MuxStatus;
         NPF_IfSONET_SDH_AdaptCurrentStatus_t   *if_AdaptStatus;
         NPF_IfSONET_SDH_MapCurrentStatus_t     *if_MapStatus;
         /* Performance Monitoring Statistics */
         NPF_IfSONET_SDH_SectionStatistics_t    *if_SectionStatistics;
```

```
        NPF_IfSONET_SDH_LineStatistics_t *if_LineStatistic;
        NPF_IfSONET_SDH_PathStatistics_t *if_MapStatistics


#endif /* temporary_no_core */

/*
** End of Asynchronous Response types for SONET/SDH
*/



/*
** --- SONET/SDH Error Codes
*/

/*
** Macro for Error Code generation
*/
#define NPF_IF_E_SONET_SDH_CODE(code)
(0x10000+(NPF_IF_TYPE_SONET_SDH<<8)+(code))



/*
** Medium
*/

/* Invalid SONET/SDH Medium Coding */
#define NPF_IF_E_INVALID_MEDIUM_CODING  NPF_IF_E_SONET_SDH_CODE(1)

/* Invalid SONET/SDH Medium Line Type */
#define NPF_IF_E_INVALID_MEDIUM_LINETYPE NPF_IF_E_SONET_SDH_CODE(2)

/* Invalid SONET/SDH Medium Loopback Configuration */
#define NPF_IF_E_INVALID_MEDIUM_LOOPBACK_CONFIG
NPF_IF_E_SONET_SDH_CODE(3)

/* Invalid SONET/SDH Medium SES Threshold Set */
#define NPF_IF_E_INVALID_MEDIUM_SES_THRESHOLDSET
NPF_IF_E_SONET_SDH_CODE(4)

/* Invalid SONET/SDH Medium Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_MEDIUM_FM_CONF NPF_IF_E_SONET_SDH_CODE(5)



/*
** Section
*/

/* Invalid SONET/SDH Section Interface Attribute */
#define NPF_IF_E_INVALID_SECTION_ATTR NPF_IF_E_SONET_SDH_CODE(6)
```

```
/* Invalid SONET/SDH SECTION Type */
#define NPF_IF_E_INVALID_SECTION _TYPE NPF_IF_E_SONET_SDH_CODE(7)


/* Invalid SONET/SDH Section Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_SECTION_FM_CONF NPF_IF_E_SONET_SDH_CODE(8)


/*
** Multiplexing
*/
/* Invalid SONET/SDH Multiplexing Interface Attribute */
#define NPF_IF_E_INVALID_MUX_ATTR NPF_IF_E_SONET_SDH_CODE(9)

/* Invalid SONET/SDH Multiplexing Interface Configuration Parameter */
#define NPF_IF_E_INVALID_MUX_FM_CONF NPF_IF_E_SONET_SDH_CODE(10)

/*
** Adaptation
*/

/* Invalid SONET/SDH Adaptation Interface Attribute */
#define NPF_IF_E_INVALID_ADAPT_ATTR NPF_IF_E_SONET_SDH_CODE(11)

/* Invalid SONET/SDH Adaptation Interface Configuration Parameter */
#define NPF_IF_E_INVALID_ADAPT_FM_CONF NPF_IF_E_SONET_SDH_CODE(12)

/*
** Mapping
*/

/* Invalid SONET/SDH MappingInterface Attribute */
#define NPF_IF_E_INVALID_MAP_ATTR NPF_IF_E_SONET_SDH_CODE(13)


/* Invalid SONET/SDH Mapping Interface Fault Management Parameter */
#define NPF_IF_E_INVALID_MAP_FM_CONFIG NPF_IF_E_SONET_SDH_CODE(14)

/*
** All SONET/SDH parent-child relationship
*/


/* Invalid SONET/SDH Interface Binding */
#define NPF_IF_E_INVALID_BINDING NPF_IF_E_SONET_SDH_CODE(15)

/*
** --- End of SONET/SDH Error Codes
*/


/*
** +++ SONET/SDH Interface Management Fault Management Events
*/
/*
** SONET/SDH Physical Medium Events
```

```
*/
#define NPF_IF_SONET_SDHMedium_LOF ((NPF_IF_TYPE_SONET_SDH<<16)+1)

          /* Medium Loss of Frame Failure */
#define NPF_IF_SONET_SDH_Medium_LOS ((NPF_IF_TYPE_SONET_SDH<<16)+1)
          /* Medium Loss of Signal */
/*
** SONET/SDH Section Events
*/
#define NPF_IF_SONET_SDH_Section_TIM ((NPF_IF_TYPE_SONET_SDH<<16)+1)
          /* Section Loss of Frame Failure */
#define NPF_IF_SONET_SDH_Section_SSF ((NPF_TYPE_SONET_SDH<<16)+ 13)
          /* Section Loss of Signal */

/*
**  SONET-SDH Multiplexing Interface (line Multiplexing Section) Events
*/
#define NPF_IF_SONET_SDH_Line_AIS ((NPF_IF_TYPE_SONET_SDH<<16)+1)
          /* Multiplexing Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Line_RDI ((NPF_TYPE_SONET_SDH<<16)+ 15)
/* Multiplexing Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Line_SSF ((NPF_TYPE_SONET_SDH<<16)+ 16)
          /* Multiplexing Interface Server Signal Failure */
          /*
          */
#define NPF_IF_SONET_SDH_Line_EXC ((NPF_TYPE_SONET_SDH<<16)+ 17)
/* Multiplexing Interface Excessive Bit Error Rate */
#define NPF_IF_SONET_SDH_Line_DEC ((NPF_TYPE_SONET_SDH<<16)+ 18)
          /* Multiplexing Interface Signal Degrade */

/*
** SONET/SDH Adaptation Interface Events
*/
#define NPF_IF_SONET_SDH_Adapt_AIS ((NPF_TYPE_SONET_SDH<<16)+ 19)
          /* Adaptation Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Adapt_RDI ((NPF_TYPE_SONET_SDH<<16)+ 20)
          /* Adaptation Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Adapt_SSF ((NPF_TYPE_SONET_SDH<<16)+ 21)
          /* Adaptation Interface Server Signal Failure */
#define NPF_IF_SONET_SDH_Adapt_LOP ((NPF_TYPE_SONET_SDH<<16)+ 22)
          /* Adaptation Interface Loss of Pointer */
/*
** SONET/SDH Mapping Interface Events
*/
#define NPF_IF_SONET_SDH_Map_AIS ((NPF_TYPE_SONET_SDH<<16)+ 23)
          /* Mapping Interface Alarm Indication Signal */
#define NPF_IF_SONET_SDH_Map_RDI ((NPF_TYPE_SONET_SDH<<16)+ 24)
          /* Mapping Interface Remote Defect Indication */
#define NPF_IF_SONET_SDH_Map_SSF ((NPF_TYPE_SONET_SDH<<16)+ 25)
          /* Mapping Interface Server Signal Failure */
          /*
          */
#define NPF_IF_SONET_SDH_Map_EXC ((NPF_TYPE_SONET_SDH<<16)+ 26)
          /* Mapping Interface Excessive Bit Error Rate */
#define NPF_IF_SONET_SDH_Map_DEC ((NPF_TYPE_SONET_SDH<<16)+ 27)
          /* Mapping Interface Signal Degrade */
```

```
           /*
           */
#define NPF_IF_SONET_SDH_Map_RFI ((NPF_TYPE_SONET_SDH<<16)+ 28)
           /* Mapping Interface Remote Failure Indication */
      /*
      */
#define NPF_IF_SONET_SDH_Map_UNEQ ((NPF_TYPE_SONET_SDH<<16)+ 29)
           /* Mapping Interface Status Unequipped */
#define NPF_IF_SONET_SDH_Map_TIM ((NPF_TYPE_SONET_SDH<<16)+ 30)
           /* Mapping Interface TTI Mismatch Defect */
#define NPF_IF_SONET_SDH_Map_PLM ((NPF_TYPE_SONET_SDH<<16)+ 31)
           /* Mapping Interface Payload Mismatch Defect */
#define NPF_IF_SONET_SDH_Map_LOM ((NPF_TYPE_SONET_SDH<<16)+ 32)
           /* Mapping Interface Loss of Multi-Frame */


/*
**  SONET-SDH Performance Monitoring Events
*/


#define NPF_IF_SONET_SDH_ES ((NPF_TYPE_SONET_SDH<<16) + 100 + 4)
           /* Errored Seconds */
#define NPF_IF_SONET_SDH_SES ((NPF_TYPE_SONET_SDH<<16) + 100 + 5)
           /* Severe Errored Seconds */
#define NPF_IF_SONET_SDH_SEFS ((NPF_TYPE_SONET_SDH<<16) + 100 + 6)
           /* Severely Errored Framing Seconds */
#define NPF_IF_SONET_SDH_CV ((NPF_TYPE_SONET_SDH<<16) + 100 + 7)
           /* Coding Violation */
#define NPF_IF_SONET_SDH_US ((NPF_TYPE_SONET_SDH<<16) + 100 +8)
           /* Unavailable Seconds */


/*
** --- End of SONET/SDH Interface Management Definitions of Events
*/


/*
** *** End of SONET/SDH Data Structure Definitions
*/



/*
** --- SONET/SDH Function Calls Definitions
*/

/*
** +++ SONET/SDH Attribute Setting Function Calls
*/

/*
** +++ Setting SONET/SDH Fault Management Configuration Parameters
*/
```

```
/*
** Set Medium Interface Fault Management Configuration Parameters
*/
NPF_error_t  NPF_IfSONET_SDH_MediumFMConfigSet (
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_IfSONET_SDH_MediumFMConfig_t
*if_SONET_SDH_MediumFMConfArray);
/*
** Set Section Interface Fault Management Configuration Parameters
*/

NPF_error_t  NPF_IfSONET_SDH_SectionFMConfigSet (
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_IfSONET_SDH_Section_FMConf_t
*if_SONET_SDH_SectionFMConfArray);
/*
** Set Multiplexing Interface Fault Management Configuration Parameters
*/

NPF_error_t  NPF_IfSONET_SDH_MuxFMConfigSet (
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_IfSONET_SDH_Mux_FMConf_t
*if_SONET_SDH_MuxFMConfArray);
/*
** Set Adaptation Interface Fault Management Configuration Parameters
*/

NPF_error_t  NPF_IfSONET_SDH_AdaptFMConfigSet (
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_IfSONET_SDH_Adapt_FMConf_t
*if_SONET_SDH_AdaptFMConfArray);
/*
** Set Mapping Interface Fault Management Configuration Parameters
*/

NPF_error_t  NPF_IfSONET_SDH_MapFMConfigSet (
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_IfSONET_SDH_Map_FMConf_t
*if_SONET_SDH_MapFMConfArray);
/*
** +++ SONET/SDH Interface Fault Management Status Getting Function
Calls
*/

/*
** Get Medium Interface Fault Management Status
*/

NPF_error_t  NPF_IfSONET_SDH_MediumStatusGet(
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);
/*
** Get Medium Interface Fault Management Current Status
*/
NPF_error_t  NPF_IfSONET_SDH_MediumCurrStatusGet(
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);
/*
** Get Section Interface Fault Management Status
*/
NPF_error_t  NPF_IfSONET_SDH_SectionStatusGet(
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);
/*
** Get Multiplexing Interface Fault Management Status
*/
NPF_error_t  NPF_IfSONET_SDH_MuxStatusGet(
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);
/*
** Get Adaptation Interface Fault Management Status
*/
NPF_error_t  NPF_IfSONET_SDH_AdaptStatusGet(
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);
/*
** Get Mapping Interface Fault Management Status
*/
NPF_error_t  NPF_IfSONET_SDH_MapStatusGet(
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);

/*
** +++ End of SONET/SDH Fault Management Function Calls
*/
/*
** +++ SONET/SDH Interface Performance Monitoring Configuration
**  Function Calls
*/

/*
** Set Medium Interface Severely Errored Signal Threshold
*/
NPF_error_t  NPF_IfSONET_SDH_MediumSESthreshAttrSet (
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_IfSONET_SDH_MediumSESthresholdSet_t
*if_SON_ET_SDH_MediumSESthreshSetArray);

/*
** Query SONET/SDH Section Interface Performance Monitoring Statistics
*/
NPF_error_t  NPF_IfSONET_SDH_SectionStatisticsQuery(
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);
/*
** Query SONET/SDH Multiplexing Interface Performance Monitoring
Statistics
*/
NPF_error_t  NPF_IfSONET_SDH_MuxStatisticsQuery(
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);
/*
** Query SONET/SDH Mapping Interface Performance Monitoring Statistics
*/
NPF_error_t  NPF_IfSONET_SDH_MapStatisticsQuery(
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);

/*
** --- End of SONET/SDH Function calls definitions
*/
```

```
#ifdef __cplusplus
}
#endif

#endif

 /* __NPF_IF_SONET_SDH_H_ */
```

Appendix B List of NPF member companies 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)
Shakti Singh (Transwitch)