Interface Management API Implementation Agreement (Specific Function Set for LAN Interfaces)

Revision 3.0

Editor: John Renwick, Agere Systems, jrenwick@agere.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 1.0.

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

Table of Contents

Table of Figures

1 Revision History

Revision	Date	Reason for Changes
3.0	11/22/2004	Extracted LAN interface features from Interface Management API Implementation Agreement version 2.0, and added editorial changes as well as technical changes as needed to make the header file independent of the other Interface Management definitions.

2 Introduction

This document defines the generic LAN interface type under the NPF Interface Management API.

2.1 Assumptions and External Requirements

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

2.2 Scope

This document is concerned only with definitions and functions supporting LAN interfaces (Ethernet, FDDI) under NPF Interface Management.

2.3 Dependencies

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

2.4 LAN Interface Management Structures

This interface type represents all broadcast media types that use IEEE MAC addressing: all the Ethernet flavors, FDDI, etc. Its application-settable attributes are:

- Port number. This is a 32-bit value whose internal structure is implementation-dependent. It identifies the physical location of the LAN connector in terms of the system view: by chassis number and board number, for example.
- Promiscuous Mode flag (promiscuous mode means receiving all frames, regardless of destination MAC address).
- LAN speed selection or autoselect enable code.
- Controls for half/full duplex, flow control.

3 Data Types

3.1 LAN Interface Data Types

3.1.1 Interface Type Code (NPF_IF_TYPE_LAN)

The type code for LAN interfaces is 2 (this is a restatement of the existing definition in the Interface Management API Implementation Agreement, Core Function Set).

```
#define NPF_IF_TYPE_LAN       2           /* LAN Interface */
```

3.1.2 LAN Speed Codes: NPF_IfLAN_Speed_t

```
/*
 * LAN Speed codes for setting Ethernet speed.
 */
typedef enum      {
      NPF_IF_LAN_SPEED_10M = 1,           /* 10 megabits/second */
      NPF_IF_LAN_SPEED_100M = 2,          /* 100 megabits/second */
      NPF_IF_LAN_SPEED_1G = 3,            /* 1 gigabit/second */
      NPF_IF_LAN_SPEED_10G = 4,           /* 10 gigabit/second */
      NPF_IF_LAN_SPEED_AUTO = 5           /* Set autosense */
} NPF_IfLAN_Speed_t;
```

3.1.3 Generic LAN Interface Attributes: NPF_IfLAN_t

A forward reference to this structure must be added to the **`NPF_IfGeneric_t`** structure in **`npf_if_core.h`** if LAN interfaces are supported. Before the declaration of **`NPF_IfGeneric_t`**, the following must appear:

```
typedef struct NPF_IfLAN        NPF_IfLAN_t;
```

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

```
          NPF_IfLAN_t *LAN_Attr;        /* LAN interface attributes */
```

```
/*
 *    LAN Control Flags
 */
#define NPF_IF_LAN_FDX_ENABLE       0x01  /* TRUE to enable full duplex */
#define NPF_IF_LAN_PROMISC          0x02  /* TRUE for promiscuous mode */
#define NPF_IF_LAN_TX_FC_ENABLE     0x04  /* TRUE for transmit flow ctrl */
#define NPF_IF_LAN_RX_FC_ENABLE     0x08  /* TRUE for receive flow ctrl */

/*
 *    Generic LAN Interface attributes
 */
struct NPF_IfLAN {
      NPF_uint32_t      port;             /* Port number */
      NPF_IfLAN_Speed_t speed;            /* Speed control */
      NPF_uint32_t      flags;            /* Flags (see above) */
      NPF_MAC_Address_t srcAddr;          /* Source MAC address */
      NPF_uint32_t      nAddrs;           /* Number of receive MAC addrs */
      NPF_MAC_Address_t *rcvAddrs;        /* Array of receive MAC addrs */
```

```
};
```

3.2 Data Structures for Completion Callbacks

3.2.1 Completion Callback Type Codes: NPF_IfCallbackType_t

```
/*
 *    Completion Callback Types for LAN interfaces
 */
#define NPF_IF_LAN_SRC_ADDR_GET                 ((NPF_IF_TYPE_LAN<<16)+1)
#define NPF_IF_LAN_SRC_ADDR_SET                 ((NPF_IF_TYPE_LAN<<16)+2)
#define NPF_IF_LAN_ADDR_LIST_SET                ((NPF_IF_TYPE_LAN<<16)+3)
#define NPF_IF_LAN_ADDR_LIST_ADD                ((NPF_IF_TYPE_LAN<<16)+4)
#define NPF_IF_LAN_ADDR_LIST_DELETE             ((NPF_IF_TYPE_LAN<<16)+5)
#define NPF_IF_LAN_PROMISC_SET                  ((NPF_IF_TYPE_LAN<<16)+6)
#define NPF_IF_LAN_PROMISC_CLEAR                ((NPF_IF_TYPE_LAN<<16)+7)
#define NPF_IF_LAN_FULL_DUPLEX_SET              ((NPF_IF_TYPE_LAN<<16)+8)
#define NPF_IF_LAN_FULL_DUPLEX_CLEAR            ((NPF_IF_TYPE_LAN<<16)+9)
#define NPF_IF_LAN_SPEED_SET                    ((NPF_IF_TYPE_LAN<<16)+10)
#define NPF_IF_LAN_FLOW_CONTROL_TX_ENABLE       ((NPF_IF_TYPE_LAN<<16)+11)
#define NPF_IF_LAN_FLOW_CONTROL_TX_DISABLE      ((NPF_IF_TYPE_LAN<<16)+12)
#define NPF_IF_LAN_FLOW_CONTROL_RX_ENABLE       ((NPF_IF_TYPE_LAN<<16)+13)
#define NPF_IF_LAN_FLOW_CONTROL_RX_DISABLE      ((NPF_IF_TYPE_LAN<<16)+14)
```

3.2.2 Asynchronous Response Details

Asynchronous responses are returned by the mechanism defined in Interface Management API Core Function Set. The NPF_IfAsyncResponse_t structure defined there contains a (void *) pointer; in the case of LAN API function calls, this pointer can point to a structure containing LAN-specific or generic information. The following table indicates what is returned by each of the functions defined in this Implementation Agreement.

Function Name	Type Code	Structure Returned
NPF_IfLAN_SrcAddrGet	NPF_IF_LAN_SRC_ADDR_GET	NPF_MAC_Address_t *
NPF_IfLAN_SrcAddrSet	NPF_IF_LAN_SRC_ADDR_SET	Unused (null pointer)
NPF_IfLAN_MAC_RcvAddrListSet	NPF_IF_LAN_ADDR_LIST_SET	Unused (null pointer)
NPF_IfLAN_MAC_RcvAddrListAdd	NPF_IF_LAN_ADDR_LIST_ADD	Unused (null pointer)
NPF_IfLAN_MAC_RcvAddrListDelete	NPF_IF_LAN_ADDR_LIST_DELETE	Unused (null pointer)
NPF_IfLAN_PromiscSet	NPF_IF_LAN_PROMISC_SET	Unused (null pointer)
NPF_IfLAN_PromiscClear	NPF_IF_LAN_PROMISC_CLEAR	Unused (null pointer)
NPF_IfLAN_FullDuplexSet	NPF_IF_LAN_FULL_DUPLEX_SET	Unused (null pointer)
NPF_IfLAN_FullDuplexClear	NPF_IF_LAN_FULL_DUPLEX_CLEAR	Unused (null pointer)
NPF_IfLAN_SpeedSet	NPF_IF_LAN_SPEED_SET	Unused (null pointer)
NPF_IfLAN_FlowControlTxEnable	NPF_IF_LAN_FLOW_CONTROL_TX_ENABLE	Unused (null pointer)
NPF_IfLAN_FlowControlTxDisable	NPF_IF_LAN_FLOW_CONTROL_TX_DISABLE	Unused (null pointer)
NPF_IfLAN_FlowControlRxEnable	NPF_IF_LAN_FLOW_CONTROL_RX_ENABLE	Unused (null pointer)
NPF_IfLAN_FlowControlRxDisable	NPF_IF_LAN_FLOW_CONTROL_RX_DISABLE	Unused (null pointer)

3.3 Error Codes

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

```
#define NPF_IF_E_LAN_CODE(code) (0x10000+(NPF_IF_TYPE_LAN<<8)+(code))

/*
 *    Asynchronous error codes (returned in function callbacks)
 *    used by LAN interface functions
 */

/* Invalid MAC address */
#define NPF_IF_E_INVALID_MAC_ADDR NPF_IF_E_LAN_CODE(1)

/* Interface has no source addr. */
#define NPF_IF_E_NO_SRC_ADDRESS NPF_IF_E_LAN_CODE(2)

/* Invalid Port number */
#define NPF_IF_E_INVALID_PORT_NUMBER NPF_IF_E_LAN_CODE(3)

/* Invalid Promiscuous Mode code */
#define NPF_IF_E_INVALID_PROMISC_MODE NPF_IF_E_LAN_CODE(4)

/* Invalid LAN speed code */
#define NPF_IF_E_INVALID_LAN_SPEED NPF_IF_E_LAN_CODE(5)
```

3.4 Data Structures for Event Notifications

3.4.1 Event Types: NPF_IfEvent_t

```
/*
 *    Event types
 */
```

4 Functions

4.1 Interface Management API

This section will define functions for querying and modifying the interface properties and attributes.

4.1.1 NPF_IfLAN_SrcAddrGet: Return a LAN Interface's Source MAC Address

Syntax

```
NPF_error_t  NPF_IfLAN_SrcAddrGet(
      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

This function returns, via a callback, the Source MAC address of a LAN interface. The Source MAC address is the address to be sent by default as Source Address in packets sent by the interface, and the default Destination Address to "listen" for in received packets.

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 the MAC address 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, or is not a LAN interface.
- **`NPF_IF_E_NO_SRC_ADDRESS`**: No source address is present in this interface.

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. If the code indicates success, the union in the callback response structure contains the Source MAC address of the interface.

4.1.2 NPF_IfLAN_SrcAddrSet: Set a LAN Interface's Source MAC Address

Syntax

```
NPF_error_t  NPF_IfLAN_SrcAddrSet(
```

```
      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_MAC_Address_t     *if_LAN_SrcAddrArray);
```

Description

This function sets the Source MAC address of one or more LAN interfaces. If more than one LAN interface is specified, each receives its MAC address from a different element of the source address array. The Source MAC address is the address to be sent by default as Source Address in packets sent by the interface, and the default Destination Address to "listen" for in received packets.

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 MAC addresses for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- **`if_LAN_SrcAddrArray`**: pointer to an array of MAC addresses containing one address for each interface handle.

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 LAN interface.
- **`NPF_IF_E_INVALID_MAC_ADDR`**: Source MAC address is 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 and a success code or a possible error code for that interface. The union in the callback response structure is unused.

4.1.3 NPF_IfLAN_MAC_RcvAddrListSet: Set a LAN Interface's List of Receive MAC Addresses

Syntax

```
NPF_error_t  NPF_IfLAN_MAC_RcvAddrListSet(
      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_uint32_t           if_n_MAC_Addrs,
      NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);
```

Description

This function sets a list of receive MAC addresses (represented as an array) for one or more interfaces. The given list replaces any list already present in the interface. If the **`if_n_MAC_Addrs`** parameter is zero, the entire list is deleted. The same address list is applied to all indicated interfaces.

Note: it is not necessary to include the Source MAC Address in an interface's list of receive MAC addresses; LAN interfaces should receive on this address by default, and the list of receive MAC addresses is in addition to the source address.

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 address list for.
- **`if_HandleArray`**: pointer to an array of interface handles.
- **`if_n_MAC_Addrs`**: number of MAC addresses in the array
- **`if_LAN_AddrArray`**: pointer to the array of receive MAC addresses to set on all the interfaces.

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 LAN interface.
- **`NPF_IF_E_INVALID_MAC_ADDR`**: One or more of the MAC addresses is invalid for receiving.

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.

4.1.4 NPF_IfLAN_MAC_RcvAddrListAdd: Add to a LAN Interface's List of Receive MAC Addresses

Syntax

```
NPF_error_t  NPF_IfLAN_MAC_RcvAddrListAdd(
      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_uint32_t           if_n_MAC_Addrs,
      NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);
```

Description

This function adds addresses to a list of receive MAC addresses (represented as an array) in one or more interfaces. The same list of addresses is added to each interface.

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 add the addresses to.
- **if_HandleArray**: the handle of the interface.
- **if_n_MAC_Addrs**: number of MAC addresses in the array
- **if_LAN_AddrArray**: pointer to an array of receive MAC addresses to add to all the indicated interfaces.

Output Parameters

None

Asynchronous Error Codes

- **NPF_NO_ERROR**: Operation successful
- **NPF_IF_E_INVALID_HANDLE**: an **if_Handle** is null or invalid, or not an LAN interface.
- **NPF_IF_E_INVALID_MAC_ADDR**: One or more of the MAC addresses is invalid for receiving.

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.

4.1.5 NPF_IfLAN_MAC_RcvAddrListDelete: Delete Receive MAC Addresses

Syntax

```
NPF_error_t  NPF_IfLAN_MAC_RcvAddrListDelete(
      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_uint32_t           if_n_MAC_Addrs,
      NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);
```

Description

This function deletes one or more addresses from a list of receive MAC addresses (represented as an array) in one or more interfaces. The same list of addresses is deleted from each interface. If an address does not exist in an interface's address list, no action is taken for that address, and no error is returned.

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 delete the addresses from.
- **if_HandleArray**: the handle of the interface.
- **if_n_MAC_Addrs**: number of MAC addresses in the array
- **if_LAN_AddrArray**: pointer to an array of receive MAC addresses to delete from all the indicated interfaces.

Output Parameters

None

Asynchronous Error Codes

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

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.

4.1.6 NPF_IfLAN_PromiscSet: Set a LAN Interface in Promiscuous Mode

Syntax

```
NPF_error_t  NPF_IfLAN_PromiscSet(
      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

This function puts one or more LAN interfaces in "promiscuous" mode (i.e. receives all packets, regardless of the destination MAC address).

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 in the **if_HandleArray** array.
- **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, or is not a LAN interface.

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.

4.1.7 NPF_IfLAN_PromiscClear: Turn off Promiscuous Mode in a LAN Interface

Syntax

```
NPF_error_t  NPF_IfLAN_PromiscClear(
      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

This function reverses the effect of **`NPF_IfLAN_PromiscSet()`** on one or more 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 in the **`if_Handle`** array.
- **`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, or is not a LAN interface.

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.

4.1.8 NPF_IfLAN_FullDuplexSet: Set Full Duplex Mode on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FullDuplexSet(
      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

This function sets Full Duplex Mode on one or more LAN interfaces. It is meaningful on 10/100 megabit Ethernet interfaces, where this option is selectable. Implementation of this function is optional.

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 in the **`if_HandleArray`** array.
- **`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, or is not a LAN interface.

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.

4.1.9 NPF_IfLAN_FullDuplexClear: Disable Full Duplex Mode on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FullDuplexClear(
      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

This function reverses the effect of NPF_IfLAN_FullDuplexSet() on one or more LAN interfaces. It is meaningful on 10/100 megabit Ethernet interfaces, where this option is selectable. Implementation of this function is optional.

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 in the **if_HandleArray** array.
- **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, or is not a LAN interface.

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.

4.1.10 NPF_IfLAN_SpeedSet: Set Interface Speed or Autosense on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_SpeedSet(
      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_IfLAN_Speed_t     *speedArray);
```

Description

This function sets the speed (10 or 100 megabit/second) or Autosense on one or more LAN interfaces. It is meaningful on 10/100 megabit Ethernet interfaces, where this option is selectable. Implementation of this function is optional.

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 in the **`if_HandleArray`** array.
- **`if_HandleArray`**: pointer to an array of interface handles.
- **`speedArray`**: pointer to an array of LAN Speed codes, one for each interface in the **`if_HandleArray`** array.

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 LAN interface.
- **`NPF_IF_E_INVALID_SPEED`**: One of the codes in the **`speedArray`** array is not valid.

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.

4.1.11 NPF_IfLAN_FlowControlTxEnable: Enable Sending Flow Control on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FlowControlTxEnable(
      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

This function enables sending flow control messages on one or more LAN interfaces. It is meaningful on 10/100 megabit interfaces in full duplex mode, or on gigabit Ethernet interfaces. Implementation of this function is optional.

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 in the **if_HandleArray** array.
- **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, or is not a LAN interface.

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.

4.1.12 NPF_IfLAN_FlowControlTxDisable: Disable Sending Flow Control on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FlowControlTxDisable(
      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

This function disables the sending of flow control messages on one or more LAN interfaces. It is meaningful on 10/100 megabit interfaces in full duplex mode, or on gigabit Ethernet interfaces. Implementation of this function is optional.

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 in the **if_HandleArray** array.
- **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, or is not a LAN interface.

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.

4.1.13 NPF_IfLAN_FlowControlRxEnable: Enable Receiving Flow Control on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FlowControlRxEnable(
      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

This function enables responding to received flow control messages on one or more LAN interfaces. It is meaningful on 10/100 megabit interfaces in full duplex mode, or on gigabit Ethernet interfaces. Implementation of this function is optional.

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 in the **`if_HandleArray`** array.
- **`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, or is not a LAN interface.

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.

4.1.14 NPF_IfLAN_FlowControlRxDisable: Disable Receiving Flow Control on a LAN Interface

Syntax

```
NPF_error_t NPF_IfLAN_FlowControlRxDisable(
      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

This function disables responding to received flow control messages on one or more LAN interfaces. It is meaningful on 10/100 megabit interfaces in full duplex mode, or on gigabit Ethernet interfaces. Implementation of this function is optional.

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 in the **`if_HandleArray`** array.
- **`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, or is not a LAN interface.

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.

5 References

1 NP Forum – Software API Conventions Implementation Agreement Revision 2.0.

6 API Capabilities

This section defines the capabilities of the Interface Management API.

It summarizes the defined APIs and Events and defines the mandatory and optional features.

6.1 Optional support of specific types

The support of any specific type of interface is optional in an implementation. An implementation MAY support exclusively one type of interface, and still claim compliance to the NP Forum Interface Management API.

6.2 API Functions

Function Name	Required?
`NPF_IfLAN_SrcAddrGet()`	Only if LAN interfaces supported
`NPF_IfLAN_SrcAddrSet()`	Only if LAN interfaces supported
`NPF_IfLAN_MAC_RcvAddrListSet()`	Only if LAN interfaces supported
`NPF_IfLAN_MAC_RcvAddrListAdd()`	Only if LAN interfaces supported
`NPF_IfLAN_PromiscSet()`	Only if LAN interfaces supported
`NPF_IfLAN_PromiscClear()`	Only if LAN interfaces supported
`NPF_IfLAN_FullDuplexSet()`	No
`NPF_IfLAN_FullDuplexClear()`	No
`NPF_IfLAN_SpeedSet()`	No
`NPF_IfLAN_FlowControlTxEnable()`	No
`NPF_IfLAN_FlowControlTxDisable()`	No
`NPF_IfLAN_FlowControlRxEnable()`	No
`NPF_IfLAN_FlowControlRxDisable()`	No

6.3 API Events

Event Name	Required?

APPENDIX A HEADER FILE: NPF_IF_LAN.H

```
/*
 * This header file defines typedefs, constants, and functions
 * that apply to the NPF Interface Management API, support for
 * LAN interface type.
 */
#ifndef __NPF_IF_LAN_H__
#define __NPF_IF_LAN_H__

#ifdef __cplusplus
extern "C"  {
#endif

#define NPF_IF_TYPE_LAN 2

/*
 * LAN Speed codes for setting Ethernet speed.
 */
typedef enum      {
      NPF_IF_LAN_SPEED_10M = 1,           /* 10 megabits/second */
      NPF_IF_LAN_SPEED_100M = 2,          /* 100 megabits/second */
      NPF_IF_LAN_SPEED_1G = 3,            /* 1 gigabit/second */
      NPF_IF_LAN_SPEED_10G = 4,           /* 10 gigabit/second */
      NPF_IF_LAN_SPEED_AUTO = 5           /* Set autosense */
} NPF_IfLAN_Speed_t;

/*
 *    LAN Control Flags
 */
#define NPF_IF_LAN_FDX_ENABLE       0x01  /* TRUE to enable full duplex */
#define NPF_IF_LAN_PROMISC    0x02  /* TRUE for promiscuous mode */
#define NPF_IF_LAN_TX_FC_ENABLE     0x04  /* TRUE for transmit flow ctrl */
#define NPF_IF_LAN_RX_FC_ENABLE     0x08  /* TRUE for receive flow ctrl */

/*
 *    Generic LAN Interface attributes
 */
struct NPF_IfLAN {
      NPF_uint32_t      port;             /* Port number */
      NPF_IfLAN_Speed_t speed;            /* Speed control */
      NPF_uint32_t      flags;            /* Flags (see above) */
      NPF_MAC_Address_t srcAddr;          /* Source MAC address */
      NPF_uint32_t      nAddrs;           /* Number of receive MAC addrs */
      NPF_MAC_Address_t *rcvAddrs;        /* Array of receive MAC addrs */
};



/*
 * Completion Callback Types
 */

#define NPF_IF_LAN_SRC_ADDR_GET                 ((NPF_IF_TYPE_LAN<<16)+1)
#define NPF_IF_LAN_SRC_ADDR_SET                 ((NPF_IF_TYPE_LAN<<16)+2)
```

```
#define NPF_IF_LAN_ADDR_LIST_SET          ((NPF_IF_TYPE_LAN<<16)+3)
#define NPF_IF_LAN_ADDR_LIST_ADD          ((NPF_IF_TYPE_LAN<<16)+4)
#define NPF_IF_LAN_ADDR_LIST_DELETE       ((NPF_IF_TYPE_LAN<<16)+5)

#define NPF_IF_LAN_PROMISC_SET                  ((NPF_IF_TYPE_LAN<<16)+6)
#define NPF_IF_LAN_PROMISC_CLEAR          ((NPF_IF_TYPE_LAN<<16)+7)
#define NPF_IF_LAN_FULL_DUPLEX_SET        ((NPF_IF_TYPE_LAN<<16)+8)
#define NPF_IF_LAN_FULL_DUPLEX_CLEAR            ((NPF_IF_TYPE_LAN<<16)+9)
#define NPF_IF_LAN_SPEED_SET              ((NPF_IF_TYPE_LAN<<16)+10)
#define NPF_IF_LAN_FLOW_CONTROL_TX_ENABLE ((NPF_IF_TYPE_LAN<<16)+11)
#define NPF_IF_LAN_FLOW_CONTROL_TX_DISABLE      ((NPF_IF_TYPE_LAN<<16)+12)
#define NPF_IF_LAN_FLOW_CONTROL_RX_ENABLE ((NPF_IF_TYPE_LAN<<16)+13)
#define NPF_IF_LAN_FLOW_CONTROL_RX_DISABLE      ((NPF_IF_TYPE_LAN<<16)+14)


/*
 *    Asynchronous error codes (returned in function callbacks)
 */

/* Invalid MAC address */
#define NPF_IF_E_INVALID_MAC_ADDR ((NPF_IfErrorType_t)
NPF_INTERFACES_BASE_ERR+13)

/* Invalid Promiscuous Mode code */
#define NPF_IF_E_INVALID_PROMISC_MODE ((NPF_IfErrorType_t)
NPF_INTERFACES_BASE_ERR+23)

/* Invalid LAN speed code */
#define NPF_IF_E_INVALID_LAN_SPEED ((NPF_IfErrorType_t)
NPF_INTERFACES_BASE_ERR+24)


NPF_error_t  NPF_IfLAN_SrcAddrGet(
    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_error_t  NPF_IfLAN_SrcAddrSet(
    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_MAC_Address_t     *if_LAN_SrcAddrArray);

NPF_error_t  NPF_IfLAN_MAC_RcvAddrListSet(
    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_uint32_t       if_n_MAC_Addrs,
    NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);

NPF_error_t  NPF_IfLAN_MAC_RcvAddrListAdd(
```

```
    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_uint32_t             if_n_MAC_Addrs,
    NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);

NPF_error_t  NPF_IfLAN_MAC_RcvAddrListDelete(
      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_uint32_t           if_n_MAC_Addrs,
      NPF_IN NPF_MAC_Address_t     *if_LAN_AddrArray);

NPF_error_t  NPF_IfLAN_PromiscSet(
    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_error_t  NPF_IfLAN_PromiscClear(
    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_error_t NPF_IfLAN_FullDuplexSet(
    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_error_t NPF_IfLAN_FullDuplexClear(
    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_error_t NPF_IfLAN_SpeedSet(
    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_IfLAN_Speed_t     *speedArray);

NPF_error_t NPF_IfLAN_FlowControlTxEnable(
    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_error_t NPF_IfLAN_FlowControlTxDisable(
    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_error_t NPF_IfLAN_FlowControlRxEnable(
    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_error_t NPF_IfLAN_FlowControlRxDisable(
    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);

#ifdef __cplusplus
}
#endif

#endif
```

APPENDIX B LIST OF COMPANIES BELONGING TO NPF DURING APPROVAL PROCESS

Agere Systems
Alcatel
Altera
AMCC
Analog Devices
Avici Systems
Azanda Network Devices
Cypress Semiconductor
Ericsson
Erlang Technologies
EZ Chip
Flextronics
Fujitsu Ltd.
FutureSoft
HCL Technologies
Hi/fn
IBM
IDT
Intel
IP Infusion
Kawasaki LSI
LSI Logic
Modelware
Mosaid
Motorola
NEC
NetLogic
Nokia
Paion Co., Ltd.
PMC Sierra
RadiSys
Samsung Electronics
Sandburst Corporation
Silicon & Software Systems
Silicon Access
Sony Electronics
STMicroelectronics
Sun Microsystems
Teja Technologies
TranSwitch
U4EA Group
Xelerated
Xilinx
Zettacom
ZTE