



ATM OAM Transmit LFB and Functional API Implementation Agreement

August 16, 2005
Revision 1.0

Editor:

Vedvyas Shanbhogue, Intel, vedvyas.shanbhogue@intel.com

Copyright © 2005 The Network Processing Forum (NPF). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction other than the following, (1) the above copyright notice and this paragraph must be included on all such copies and derivative works, and (2) this document itself may not be modified in any way, such as by removing the copyright notice or references to the NPF, except as needed for the purpose of developing NPF Implementation Agreements.

By downloading, copying, or using this document in any manner, the user consents to the terms and conditions of this notice. Unless the terms and conditions of this notice are breached by the user, the limited permissions granted above are perpetual and will not be revoked by the NPF or its successors or assigns.

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "AS IS" BASIS WITHOUT ANY WARRANTY OF ANY KIND. THE INFORMATION, CONCLUSIONS AND OPINIONS CONTAINED IN THE DOCUMENT ARE THOSE OF THE AUTHORS, AND NOT THOSE OF NPF. THE NPF DOES NOT WARRANT THE INFORMATION IN THIS DOCUMENT IS ACCURATE OR CORRECT. THE NPF DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED THE IMPLIED LIMITED WARRANTIES OF MERCHANTABILITY, TITLE OR FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

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

| | | |
|---|----------------------------------------------------------------------------|----|
| 1 | Revision History | 3 |
| 2 | Introduction..... | 4 |
| | 2.1 Acronyms..... | 4 |
| | 2.2 Assumptions..... | 4 |
| | 2.3 Scope | 4 |
| | 2.4 External Requirements and Dependencies..... | 4 |
| 3 | ATM OAM Transmit Description | 6 |
| | 3.1 ATM OAM Transmit Inputs..... | 7 |
| | 3.2 ATM OAM Transmit Outputs | 8 |
| | 3.3 Accepted Inputs | 8 |
| | 3.4 Cell Modifications | 8 |
| | 3.5 Relationship with Other LFBs | 9 |
| 4 | Data Types | 11 |
| | 4.1 Common LFB Data Types..... | 11 |
| | 4.2 Data Structures for Completion Callbacks | 11 |
| | 4.3 Data Structures for Event Notifications..... | 13 |
| | 4.4 Error Codes | 13 |
| 5 | Functional API (FAPI)..... | 14 |
| | 5.1 Required Functions | 14 |
| | 5.2 Conditional Functions..... | 14 |
| | 5.3 Optional Functions..... | 16 |
| 6 | References..... | 17 |
| | Appendix A Header File Information..... | 18 |
| | Appendix B Acknowledgements..... | 20 |
| | Appendix C List of companies belonging to NPF During Approval Process..... | 21 |

Table of Figures

| | | |
|-------------|--------------------------------------------------------------------|---|
| Figure 3.1: | ATM OAM Transmit LFB..... | 6 |
| Figure 3.2: | F4/F5 Flow Instances | 7 |
| Figure 3.3: | Cooperation between ATM OAM Transmit and ATM Traffic Manager | 9 |

List of Tables

| | | |
|------------|---------------------------------------------------|----|
| Table 3.1: | ATM OAM Transmit LFB Inputs | 7 |
| Table 3.2: | Input Metadata for ATM OAM Transmit LFB | 7 |
| Table 3.3: | ATM OAM Transmit LFB Outputs..... | 8 |
| Table 3.4: | Output Metadata for ATM OAM Transmit LFB | 8 |
| Table 4.1: | Callback type to callback data mapping table..... | 12 |

1 Revision History

| Revision | Date | Reason for Changes |
|----------|------------|----------------------------------------------------------------------------------------------------------|
| 1.0 | 08/03/2005 | Rev 1.0 of the ATM OAM Transmit LFB and Functional API Implementation Agreement. Source: npf2004.162.07. |

2 Introduction

This contribution defines the ATM OAM Transmit LFB and lists configurations that are required in the LFB.

2.1 Acronyms

- **AIS:** Alarm Indication Signal
- **API:** Application Program Interface
- **ATM:** Asynchronous Transfer Mode
- **BR:** Backward Reporting
- **CC:** Continuity Check
- **FAPI:** Functional API
- **F4:** OAM flow on virtual path level
- **F5:** OAM flow on virtual channel level
- **FPM:** Forward Performance Monitoring
- **ID:** Identifier
- **LB:** Loopback
- **LFB:** Logical Functional Block
- **LLID:** Loopback Location ID
- **NNI:** Network Node Interface
- **OAM:** Operation and Maintenance
- **PM:** Performance Monitoring
- **PTI:** Payload Type Indicator
- **PVC:** Permanent Virtual Connection
- **RDI:** Remote Defect Indication
- **SDU:** Service Data Unit
- **UNI:** User Network Interface
- **VC:** Virtual Connection
- **VCC:** Virtual Circuit Connection
- **VCI:** Virtual Channel Identifier
- **VPC:** Virtual Path Connection
- **VPI:** Virtual Path Identifier

2.2 Assumptions

The ATM OAM Transmit LFB obtains its configurations from the ATM Configuration Manager Functional API implementation. The mechanism used to obtain this configuration is not in the scope of NPF.

2.3 Scope

This IA describes the configurations required by the LFB for ATM OAM processing. The IA also specifies the metadata generated and consumed by this LFB.

2.4 External Requirements and Dependencies

This document depends on the following documents:

- This document depends on the NPF Software API Conventions Implementation Agreement document [SWAPICON] for basic type definitions. (Refer section 5.1 of Software API Conventions IA Revision 2.0).
- This document depends on Software API Conventions Implementation agreement Revision 2.0 for the below type definitions
 - NPF_error_t – Refer section 5.2 of Software API Conventions IA Rev 2.0
 - NPF_callbackHandle_t - Refer section 5.2 of Software API Conventions IA Rev 2.0
 - NPF_callbackType_t - Refer section 5.2 of Software API Conventions IA Rev 2.0
 - NPF_userContext_t - Refer section 5.2 of Software API Conventions IA Rev 2.0
 - NPF_errorReporting_t - Refer section 5.2 of Software API Conventions IA Rev 2.0
- This document depends on Topology Manager Functional API Implementation Agreement Revision 1.0 for the below type definitions
 - NPF_BlockId_t – Refer section 3.1.1 of Topology Manager Functional API IA Rev 1.0
 - NPF_FE_Handle_t – Refer section 3.1.1 of Topology Manager Functional API IA Rev 1.0
- ATM Software API Architecture Framework Implementation Agreement Revision 1.0 defines the architectural framework for the ATM FAPIs.
- ATM Configuration Manager Functional API Implementation Agreement Revision 1.0 defines the functions to configure and manage ATM LFBs on a forwarding element.

3 ATM OAM Transmit Description

The ATM OAM Transmit LFB does the insertion of the ATM OAM cells in the transmitted ATM cell stream. The ATM OAM cell insertion may be requested by the ATM OAM receive LFB as a response to received ATM OAM cells, detected fault conditions, performance monitoring functions performed by the ATM OAM receive LFB, etc. The ATM OAM Transmit LFB also performs OAM cell insertion as directed by the FAPI client to carry out procedures like activation or deactivation of continuity check/performance monitoring procedures, alarm generation, loopback initiation, etc.

When performance monitoring is enabled for a F4 or F5 flow, the ATM OAM transmit LFB maintains statistics for the user cells transmitted on those flows and generates FPM cells periodically based on the configured block size.

When continuity check procedures are enabled for a F4 or F5 flow, the continuity check cells are generated by the ATM OAM transmit LFB. The CC cells may be generated periodically or if no user cell for that flow has been transmitted for the continuity check period as specified by the FAPI client when activating the continuity check procedure.

The ATM OAM Transmit LFB receives ATM SDUs received from the previous LFB over the ATM_SDU_IN input. The ATM OAM transmit LFB receives the ATM SDUs for ATM OAM cells requested for insertion into the F4/F5 flows by the ATM OAM receive LFB over the OAM_TX_IN input.

OAM flows are related to bi-directional Maintenance Entities (MEs) corresponding either to the entire ATM VPC/VCC, referred to as the VPC/VCC ME, or to a portion of this connection referred to as a VPC/VCC segment ME. Before the start of any OAM operation, the boundary needs to be drawn for the paired endpoints. The MEs terminating the ATM links are configured before as an endpoint of the VPC/VCC or endpoint of the VPC/VCC segment. End-to-end F5 flows terminate at the endpoints of a VCC, while the segment F5 flows terminate at the VCC segment endpoints. Similarly, the end-to-end F4 flows terminate at the endpoints of a VPC, while the segment F4 flows terminate at the VPC segment endpoints. The ATM OAM Transmit LFB performs the OAM functions configured for the OAM flow terminations.

The ATM OAM Transmit LFB is modeled as shown in Figure 3.1

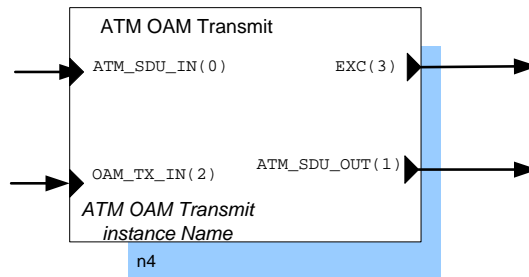


Figure 3.1: ATM OAM Transmit LFB

The LFB may contain multiple instances of F4 flows identified by unique VP Link IDs. The LFB may contain multiple instances of F5 flows identified by unique VC Link IDs. ATM cells are associated with VC links only when the VP link carrying the cell is terminated at this node. Such instances are depicted in Figure 3.2.

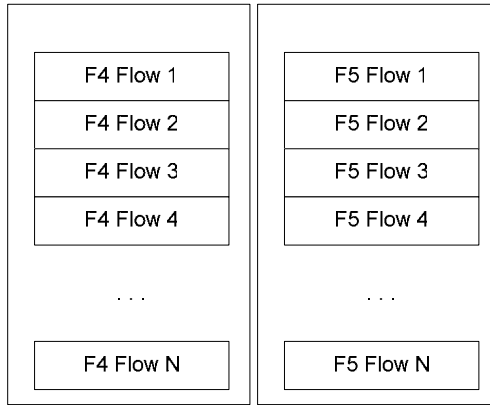


Figure 3.2: F4/F5 Flow Instances

Depending on the connection on which the cells are to be transmitted, the cells may be classified into user cells and OAM cells for the OAM flow associated with that connection. The cells to be transmitted on a VP link may be either user cells or OAM cells for F4 flow. The cells to be transmitted on a VC link may be either user cells or OAM cells for F5 flow. Additionally, F5 OAM cells transmitted on VC links are considered as user cells for the F4 flow on the associated VP. The F4 and F5 flows are bidirectional and the OAM cells for both directions of the flow must follow the same physical route so that it is possible for any CP on that connection to correlate the fault and performance information from both directions.

3.1 ATM OAM Transmit Inputs

Table 3.1: ATM OAM Transmit LFB Inputs

| Symbolic Name | Input ID | Description |
|---------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ATM_SDU_IN | 0 | This input is used to receive ATM SDUs from the previous LFB. |
| OAM_TX_IN | 2 | This input is used to receive ATM SDUs for OAM cells requested for insertions by the ATM OAM receive LFB or from any other source which requires insertion of OAM cells. |

3.1.1 Metadata Required

Table 3.2: Input Metadata for ATM OAM Transmit LFB

| Metadata tag | Access method | Description |
|--------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| META_VPL_ID | Read | Metadata identifying the VP link on which the ATM cell is to be transmitted. |
| META_VCL_ID | Read | Metadata identifying the VC link on which the ATM cell is to be transmitted. This metadata is specified only when the corresponding VP link is terminated at this node. |
| META_ATM_PTI | Read | Payload Type of ATM cell. |
| META_ATM_LP | Read | Loss priority of the ATM cell. |

| | | |
|--------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| META_ATM_VCI | Read | Metadata identifying the VCI to be used for the ATM cell to be transmitted. This metadata is expected only for VP switched links. The VP link ID identifies the VPI for the ATM cell. This metadata identifies the VCI for the cell. |
|--------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

3.2 ATM OAM Transmit Outputs

Table 3.3: ATM OAM Transmit LFB Outputs

| Symbolic Name | Output ID | Description |
|---------------|-----------|-----------------------------------------------------------------------------------------------|
| ATM_SDU_OUT | 1 | This is the normal output for the ATM OAM Transmit LFB. |
| EXC | 3 | The ATM SDUs requested for transmission are sent to this output when discarded due to errors. |

3.2.1.1 Metadata Produced

The ATM OAM Transmit LFB does not modify or generate any metadata for ATM SDU received on its input. For ATM OAM cell generation initiated by the ATM OAM Transmit LFB, the following metadata is generated.

Table 3.4: Output Metadata for ATM OAM Transmit LFB

| Metadata tag | Access method | Description |
|--------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| META_VPL_ID | Write | Metadata identifying the VP link on which the ATM OAM cell is to be transmitted. This metadata is generated when the ATM OAM cell is a F4 OAM cell. |
| META_VCL_ID | Write | Metadata identifying the VC link on which the ATM OAM cell is to be transmitted. This metadata is generated when the ATM OAM cell is a F5 OAM cell. |
| META_ATM_VCI | Write | Metadata identifying the VCI to be used for the ATM OAM cell to be transmitted. This metadata is only generated when sending F4 OAM cells. |
| META_ATM_PTI | Write | Payload Type of ATM cell. |
| META_ATM_LP | Write | Loss priority of the ATM cell. |

3.3 Accepted Inputs

The ATM OAM Transmit LFB accepts ATM SDUs for transmission over UNI or NNI interface.

3.4 Cell Modifications

- The source point of a VCC segment will discard all incoming VC segment OAM cells.
- The source point of VPC segment will discard all incoming VP segment OAM cells.

The user cells received on its input are not modified by the ATM OAM Transmit LFB. The ATM OAM Transmit LFB will not change the order of the cells input to the LFB.

3.5 Relationship with Other LFBs

The ATM OAM Transmit LFB may be placed in the processing chain before the ATM Traffic manager LFB. The ATM OAM Transmit LFB receives ATM SDUs on its inputs. The sequence of actions that configures ATM OAM Transmit LFB and cooperating ATM Traffic manager LFB instance, and cooperation between these two LFBs is schematically depicted in Figure 3.3.

The ATM OAM Transmit LFB may be preceded in the topology by any LFB that can produce the information required by the ATM OAM Transmit LFB at its input. Downstream (not necessarily next) of the ATM OAM Transmit LFB, there should be LFBs that can utilize the information generated at output by ATM OAM Transmit LFB. The exact design and connections between the ATM OAM Transmit LFB and cooperating blocks is specific to the vendor that provides Forwarding Element design and FAPI implementation.

The EXC output of the ATM OAM Transmit LFB could be connected to an LFB that receives cells for which could not be processed due to errors. Depending on system design this may be either dropper, which drops cells, or other LFB that makes a decision how to utilize such cells.

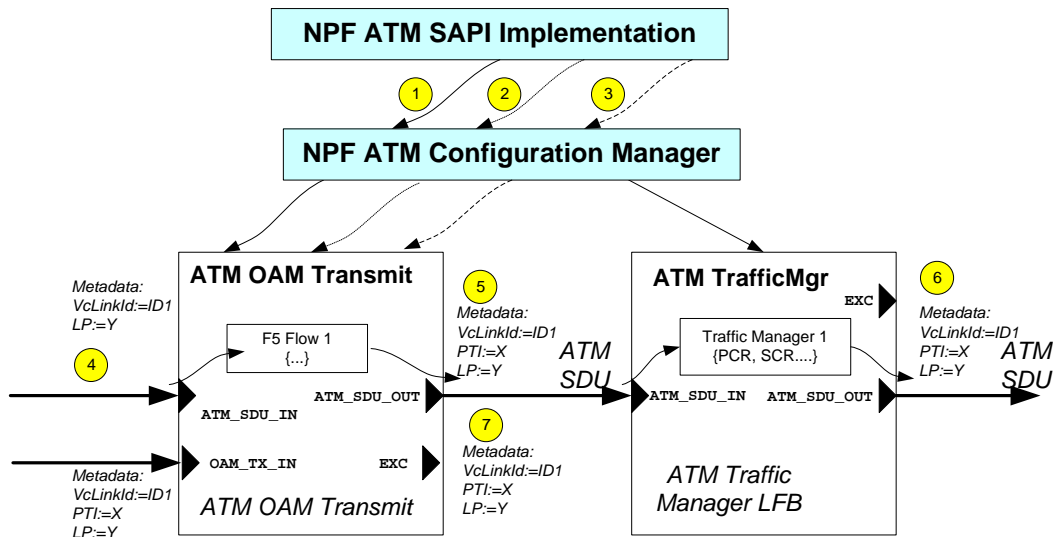


Figure 3.3: Cooperation between ATM OAM Transmit and ATM Traffic Manager

This figure shows part of example Forwarding Element that contains ATM OAM Transmit LFB and the ATM Traffic Manager LFB. These two blocks are connected in chain and configured by a NPF SAPI implementation. The sequence of actions that configure a VC link and enabling performance monitoring on the configured VC link leading to generation of an FPM cell may be defined as follows (see corresponding numbers in circles in the figure):

1. The NPF ATM SAPI is invoked to create an ATM VC link. The system software under the NPF ATM SAPI assigns a VC link ID 'ID1' to it and invokes the ATM Configuration manager FAPI to create the ATM VC link. This causes an ATM VC link instance to be created in the ATM Traffic Manager LFB and an F5 flow instance to be created in the ATM OAM Transmit LFB.
2. The NPF ATM SAPI is invoked to configure the ATM VC link as a segment endpoint for F5 flow. This system software under the NPF ATM SAPI invokes the ATM configuration manager FAPI to configure the ATM VC link as a segment endpoint for F5 flow.
3. The NPF ATM SAPI is invoked to configure FPM only performance monitoring procedures for FPM generation in A-B direction on the VC link with link ID 'ID1'. The system software under

the NPF ATM SAPI invokes the ATM configuration manager FAPI to configure performance monitoring procedure on the ATM VC link.

4. The ATM SDU is passed along with the metadata to the ATM OAM Transmit LFB. The ATM OAM Transmit LFB uses the VC link ID received in the input metadata to identify the F5 flow instance. The current cell is identified as a user cell for F5 flow and as performance monitoring is enabled for this flow the performance monitoring statistics are updated.
5. The received ATM SDU is passed to the next LFB in the processing chain over the ATM_SDU_OUT output. The input metadata is passed without any modifications.
6. The ATM Traffic manager performs traffic management actions on the received ATM SDU. The ATM SDU is sent to the next LFB in the chain for transmission on the line.
7. Transmission of this ATM SDU leads to the completion of the configured block size for performance monitoring and an ATM SDU for transmission of segment FPM OAM cell on VC link is generated by the ATM OAM Transmit LFB.

4 Data Types

4.1 Common LFB Data Types

4.1.1 LFB Type Code

It is possible to use the FAPI Topology Discovery APIs to discover an ATM OAM Transmit LFB in a forwarding element using a block type value for the ATM OAM Transmit LFB.

```
#define NPF_F_ATMOAMTX_LFB_TYPE 42
```

4.1.2 ATM OAM Transmit ATM link characteristics

The ATM OAM Transmit LFB requires below configurations for each F4/F5 flow:

- VP/VC link ID
- Connection point type – ETE endpoint, segment end point, ETE and segment endpoint, intermediate point for ETE flow, intermediate point for ETE and segment flows
- Whether LLID option is enabled
- The connection point ID
- If performance monitoring functions are enabled, then the following configurations are required
 - Performance monitoring function – FPM-BR or FPM
 - Direction – Forward, backward, two way
 - Forward direction block size (A-B direction)
 - Backward direction block size (B-A direction)
- If continuity check functions are enabled, then the following configurations are required
 - Direction – Forward, backward, two way
 - Continuity check method – whether continuity check sent periodically or sent only in the absence of user cells
- If loopback operation is to be carried out, then the following configurations are required
 - Loopback location ID
 - Whether source connection point ID to be included in the loopback cell
- AIS alarm states for the specified link. The following configurations are needed
 - Defect type
 - Defect location

4.2 Data Structures for Completion Callbacks

4.2.1 ATM OAM Transmit LFB Attributes query response

The attributes of an ATM OAM Transmit LFB are the following:

```
typedef struct {
    NPF_uint32_t    maxF4Flows;           /* Maximum possible F4 flows */
    NPF_uint32_t    curNumF4Flows;       /* Current number of F4 flows */
    NPF_uint32_t    maxF5Flows;           /* Maximum possible F5 flows */
    NPF_uint32_t    curNumF5Flows;       /* Current number of F5 flows */
    NPF_uint32_t    maxF4PMPProcess;     /* Maximum F4 PM processes */
    NPF_uint32_t    curNumF4PMPProcess;  /* Current number of F4 PM procs */
    NPF_uint32_t    maxF5PMPProcess;     /* Maximum F5 PM processes */
    NPF_uint32_t    curNumF5PMPProcess;  /* Current number of F5 PM procs */
} NPF_F_ATMOamTxLFB_AttrQueryResponse_t;
```

The `maxF4Flows` field contains the maximum number of ATM VP links supported in this ATM OAM Transmit LFB. The `curNumF4Flows` field contains the number of ATM VP links established in the LFB. The `maxF5Flows` field contains the maximum number of ATM VC links supported in this ATM OAM Receive LFB. The `curNumF5Flows` field contains the number of ATM VC links established in the LFB. The `maxF4PMPProcesses` and `maxF5PMPProcesses` fields indicate the maximum number of F4 and F5 flows on which performance monitoring processes may be activated in this LFB. The `curNumF4PMPProcesses` and `curNumF5PMPProcesses` fields indicate the current number of F4 and F5 flows on which performance monitoring processes are activated.

4.2.2 Asynchronous Response

The Asynchronous Response data structure is used during callbacks in response to API invocations.

```

/*
 * An asynchronous response contains an error or success code, and in some
 * cases a function specific structure embedded in a union.
 */
typedef struct { /* Asynchronous Response Structure */
    NPF_F_ATMOamTxErrorType_t error; /* Error code for this response*/
    union {
        /* NPF_F_ATMOamTxLFB_AttributesQuery() */
        NPF_F_ATMOamTxLFB_AttrQueryResponse_t lfbAttrQueryResponse;
    } u;
} NPF_F_ATMOamTxAsyncResponse_t;

```

4.2.3 Callback Type

This enumeration is used to indicate reason for invoking the callback function.

```

/*
 * Completion Callback Types, to be found in the callback
 * data structure, NPF_F_ATMOamTxCallbackData_t.
 */
typedef enum NPF_F_ATMOamTxCallbackType {
    NPF_F_ATMOAMTX_ATTR_QUERY = 1,
} NPF_F_ATMOamTxCallbackType_t;

```

4.2.3.1 Callback Data

An asynchronous response contains an error or success code and a function-specific structure embedded in a union in the `NPF_F_ATMOamTxCallbackData_t` structure.

```

/*
 * The callback function receives the following structure containing
 * an asynchronous responses from a function call.
 * For the completed request, the error code is specified in the
 * NPF_ATMOamTxAsyncResponse_t structure, along with any other information
 */
typedef struct {
    NPF_F_ATMOamTxCallbackType_t type; /* Which function called? */
    NPF_IN NPF_BlockId_t blockId; /* ID of LFB generating callback */
    NPF_F_ATMOamTxAsyncResp_t *resp; /* Pointer to response struct */
} NPF_F_ATMOamTxCallbackData_t;

```

The callback data that returned for different callback types is summarized in Table 4.1.

Table 4.1: Callback type to callback data mapping table

| Callback Type | Callback Data |
|----------------------------------------|----------------------------------------------------|
| <code>NPF_F_ATMOAMTX_ATTR_QUERY</code> | <code>NPF_F_ATMOamTxLFB_AttrQueryResponse_t</code> |

4.3 Data Structures for Event Notifications

4.3.1 Event Notification Types

None

4.3.2 Event Notification Structures

None

4.4 Error Codes

4.4.1 Common NPF Error Codes

The common error codes that are returned by ATM OAM Transmit LFB are listed below:

- NPF_NO_ERROR - This value MUST be returned when a function was successfully invoked. This value is also used in completion callbacks where it MUST be the only value used to signify success.
- NPF_E_UNKNOWN - An unknown error occurred in the implementation such that there is no error code defined that is more appropriate or informative.
- NPF_E_BAD_CALLBACK_HANDLE - A function was invoked with a callback handle that did not correspond to a valid NPF callback handle as returned by a registration function, or a callback handle was registered with a registration function belonging to a different API than the function call where the handle was passed in.
- NPF_E_BAD_CALLBACK_FUNCTION - A callback registration was invoked with a function pointer parameter that was invalid.
- NPF_E_CALLBACK_ALREADY_REGISTERED - A callback or event registration was invoked with a pair composed of a function pointer and a user context that was previously used for an identical registration.
- NPF_E_FUNCTION_NOT_SUPPORTED - This error value MUST be returned when an optional function call is not implemented by an implementation. This error value MUST NOT be returned by any required function call. This error value MUST be returned as the function return value (i.e., synchronously).
- NPF_E_RESOURCE_EXISTS - A duplicate request to create a resource was detected. No new resource was created.
- NPF_E_RESOURCE_NONEXISTENT - A duplicate request to destroy or free a resource was detected. The resource was previously destroyed or never existed.

4.4.2 LFB Specific Error Codes

This section defines ATM OAM Transmit Configuration and management APIs error codes. These codes are used in callbacks to deliver results of the requested operations.

```
#define NPF_ATMOAMTX_BASE_ERR (NPF_F_ATMOAMTX_LFB_TYPE * 100)
/* Asynchronous error codes (returned in function callbacks) */
typedef NPF_uint32_t NPF_F_ATMOamTxErrorType_t;

#define ATMOAMTX_ERR(n) ((NPF_F_ATMOamTxErrorType_t) \
                        (NPF_ATMOAMTX_BASE_ERR+ (n)))
/* LFB ID is not an ID of LFB that has ATM OAM Transmit functionality*/
#define NPF_E_ATMOAMTX_INVALID_ATMOAMTX_BLOCK_ID ATMOAMTX_ERR(0)
```

5 Functional API (FAPI)

5.1 Required Functions

None

5.2 Conditional Functions

The conditional API functions for registration and de-registration of the completion callback functions need to be implemented if any of the optional functions defined for this LFB are implemented.

5.2.1 Completion Callback Function

```
typedef void (*NPF_F_ATMOamTxCallbackFunc_t) (
    NPF_IN NPF_userContext_t          userContext,
    NPF_IN NPF_correlator_t          correlator,
    NPF_IN NPF_F_ATMOamTxCallbackData_t data);
```

5.2.1.1 Description

This callback function is for the application to register an asynchronous response handling routine to the ATM OAM Transmit API implementation. This callback function is intended to be implemented by the application, and be registered to the ATM OAM Transmit API implementation through the `NPF_F_ATMOamTxRegister` function.

5.2.1.2 Input Parameters

- `userContext` - The context item that was supplied by the application when the completion callback routine was registered.
- `correlator` - The correlator item that was supplied by the application when the ATM OAM Transmit API function call was invoked.
- `data` - The response information related to the particular callback type.

5.2.1.3 Output Parameters

None

5.2.1.4 Return Values

None

5.2.2 Completion Callback Registration Function

```
NPF_error_t NPF_F_ATMOamTxRegister(
    NPF_IN NPF_userContext_t          userContext,
    NPF_IN NPF_F_ATMOamTxCallbackFunc_t callbackFunc,
    NPF_OUT NPF_callbackHandle_t      *callbackHandle);
```

5.2.2.1 Description

This function is used by an application to register its completion callback function for receiving asynchronous responses related to ATM OAM Transmit API function calls. Applications MAY register multiple callback functions using this function. The pair of `userContext` and `callbackFunc` identifies the callback function. For each individual pair, a unique `callbackHandle` will be assigned for future reference. Since the callback function is identified by both `userContext` and `callbackFunc`, duplicate registration of the same callback function with a different `userContext` is allowed. Also, the same `userContext` can be shared among different callback functions. Duplicate registration of the same `userContext` and `callbackFunc` pair has no effect, and will output a handle that is already assigned to the pair, and will return `NPF_E_ALREADY_REGISTERED`.

5.2.2.2 Input Parameters

- `userContext` – A context item for uniquely identifying the context of the application registering the completion callback function. The exact value will be provided back to the registered completion callback function as its first parameter when it is called. Applications can assign any value to the `userContext` and the value is completely opaque to the API implementation.
- `callbackFunc` – The pointer to the completion callback function to be registered.

5.2.2.3 Output Parameters

- `callbackHandle` - A unique identifier assigned for the registered `userContext` and `callbackFunc` pair. This handle will be used by the application to specify which callback function to be called when invoking asynchronous NPF ATM OAM Transmit API functions. It will also be used when deregistering the `userContext` and `callbackFunc` pair.

5.2.2.4 Return Values

- `NPF_NO_ERROR` - The registration completed successfully.
- `NPF_E_BAD_CALLBACK_FUNCTION` – The `callbackFunc` is NULL, or otherwise invalid.
- `NPF_E_ALREADY_REGISTERED` – No new registration was made since the `userContext` and `callbackFunc` pair was already registered.

5.2.2.5 Notes

- This API function may be invoked by any application interested in receiving asynchronous responses for ATM OAM Transmit API function calls.
- This function operates in a synchronous manner, providing a return value as listed above.

5.2.3 Completion Callback Deregistration Function

```
NPF_error_t NPF_F_ATMOamTxDeregister(  
    NPF_IN NPF_callbackHandle_t    callbackHandle);
```

5.2.3.1 Description

This function is used by an application to deregister a user context and callback function pair.

5.2.3.2 Input Parameters

- `callbackHandle` - The unique identifier returned to the application when the completion callback routine was registered.

5.2.3.3 Output Parameters

None

5.2.3.4 Return Values

- `NPF_NO_ERROR` - De-registration was completed successfully.
- `NPF_E_BAD_CALLBACK_HANDLE` – De-registration did not complete successfully due to problems with the callback handle provided.

5.2.3.5 Notes

- This API function MAY be invoked by any application no longer interested in receiving asynchronous responses for ATM OAM Transmit API function calls.
- This function operates in a synchronous manner, providing a return value as listed above.
- There may be a timing window where outstanding callbacks continue to be delivered to the callback routine after de-registration function has been invoked. It is the implementation's responsibility to guarantee that the callback function is not called after the deregister function has returned.

5.3 Optional Functions

5.3.1 LFB Attributes Query Function

```
NPF_error_t NPF_F_ATMOamTxLFB_AttributesQuery(
    NPF_IN NPF_callbackHandle_t    callbackHandle,
    NPF_IN NPF_correlator_t        correlator,
    NPF_IN NPF_errorReporting_t    errorReporting,
    NPF_IN NPF_FE_Handle_t         feHandle,
    NPF_IN NPF_BlockId_t           blockId);
```

5.3.1.1 Description

This function call is used to query ONLY one ATM OAM Transmit LFB's attributes at a time. If the ATM OAM Transmit LFB exists, the various attributes of this LFB are returned in the completion callback.

5.3.1.2 Input Parameters

- `callbackHandle` - The unique identifier provided to the application when the completion callback routine was registered.
- `correlator` - A unique application invocation context that will be supplied to the asynchronous completion callback routine.
- `errorReporting` - An indication of whether the application desires to receive an asynchronous completion callback for this API invocation.
- `feHandle` - The FE Handle returned by `NPF_F_topologyGetFEInfoList()` call.
- `blockId` - The unique identification of the ATM OAM Transmit LFB.

5.3.1.3 Output Parameters

None

5.3.1.4 Return Values

- `NPF_NO_ERROR` - The operation is in progress.
- `NPF_E_UNKNOWN` - The LFB attributes was not queried due to invalid ATM OAM Transmit block ID passed in input parameters.
- `NPF_E_BAD_CALLBACK_HANDLE` - The LFB attributes was not queried because the callback handle was invalid.
- `NPF_E_FUNCTION_NOT_SUPPORTED` - The function call is not supported.

5.3.1.5 Asynchronous Response

There may be multiple asynchronous callbacks to this request. Possible error codes are:

- `NPF_NO_ERROR` - Operation completed successfully.
- `NPF_E_ATMOAMTX_INVALID_ATMOAMTX_BLOCK_ID` - LFB ID is not an ID of LFB that has ATM OAM Transmit functionality

The `lfbAttrQueryResponse` field of the union in the `NPF_F_ATMOamTxAsyncResponse_t` structure returned in callback contains response data. The error code is returned in the error field.

6 References

The following documents contain provisions, which through reference in this text constitute provisions of this specification. At the time of publication, the editions indicated were valid. All referenced documents are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

- [FORCESREQ] "Requirement for separation of IP control and forwarding", H.Khosravi, T.Anderson et al, November, 2003 (RFC 3654).
- [FAPITOPO] "Topology Manager Functional API", http://www.npforum.org/techinfo/topology_fapi_npf2002%20438%2023.pdf, Network Processing Forum.
- [SWAPICON] "Software API Conventions Revision 2", http://www.npforum.org/techinfo/APIConventions2_1A.pdf, Network Processing Forum.
- [ATMLFBARC] "ATM Software API Architecture Framework", <http://www.npforum.org/techinfo/npf2004.088.12.pdf>, Network Processing Forum.
- [ATMMGR] "ATM Configuration Manager Functional API", <http://www.npforum.org/techinfo/npf2004.165.31.pdf>, Network Processing Forum.

Appendix A Header File Information

```

/*
 * This header file defines typedefs, constants and structures
 * for the NP Forum ATM OAM Transmit Functional API
 */

#ifndef __NPF_F_ATM_OAMTX_H__
#define __NPF_F_ATM_OAMTX_H__

#ifdef __cplusplus
extern "C" {
#endif

/* It is possible to use the FAPI Topology Discovery
   APIs to discover an ATM OAM Transmit LFB
   in a forwarding element. */
#define NPF_F_ATMOAMTX_LFB_TYPE 42

/* Asynchronous error codes (returned in function callbacks) */
typedef NPF_uint32_t NPF_F_ATMOamTxErrorType_t;

#define NPF_ATMOAMTX_BASE_ERR (NPF_F_ATMOAMTX_LFB_TYPE * 100)
#define ATMOAMTX_ERR(n) ((NPF_F_ATMOamTxErrorType_t) \
                        (NPF_ATMOAMTX_BASE_ERR+ (n)))
/* LFB ID is not an ID of LFB that has ATM OAM Transmit functionality*/
#define NPF_E_ATMOAMTX_INVALID_ATMOAMTX_BLOCK_ID ATMOAMTX_ERR (0)

/*****
 * Enumerations and types for ATM OAM Tx attributes and
 * completion callback data types
 *****/

/* The attributes of an ATM OAM Transmit LFB */
typedef struct {
    NPF_uint32_t    maxF4Flows;           /* Maximum possible F4 flows */
    NPF_uint32_t    curNumF4Flows;       /* Current number of F4 flows */
    NPF_uint32_t    maxF5Flows;         /* Maximum possible F5 flows */
    NPF_uint32_t    curNumF5Flows;       /* Current number of F5 flows */
    NPF_uint32_t    maxF4PMPProcess;     /* Maximum F4 PM processes */
    NPF_uint32_t    curNumF4PMPProcess;  /* Current number of F4 PM procs */
    NPF_uint32_t    maxF5PMPProcess;     /* Maximum F5 PM processes */
    NPF_uint32_t    curNumF5PMPProcess;  /* Current number of F5 PM procs */
} NPF_F_ATMOamTxLFB_AttrQueryResponse_t;

/*
 * An asynchronous response contains an error or success code, and in some
 * cases a function specific structure embedded in a union.
 */
typedef struct { /* Asynchronous Response Structure */
    NPF_F_ATMOamTxErrorType_t error; /* Error code for this response*/
    union {
        /* NPF_F_ATMOamTxLFB_AttributesQuery() */
        NPF_F_ATMOamTxLFB_AttrQueryResponse_t    lfbAttrQueryResponse;
    } u;
} NPF_F_ATMOamTxAsyncResponse_t;

/*
 * Completion Callback Types, to be found in the callback

```

```

* data structure, NPF_F_ATMOamTxCallbackData_t.
*/
typedef enum NPF_F_ATMOamTxCallbackType {
    NPF_F_ATMOAMTX_ATTR_QUERY = 1,
} NPF_F_ATMOamTxCallbackType_t;

/*
* The callback function receives the following structure containing
* an asynchronous responses from a function call.
* For the completed request, the error code is specified in the
* NPF_ATMOamTxAsyncResponse_t structure, along with any other information
*/
typedef struct {
    NPF_F_ATMOamTxCallbackType_t type; /* Which function called? */
    NPF_IN NPF_BlockId_t blockId; /*ID of LFB generating callback */
    NPF_F_ATMOamTxAsyncResponse_t resp; /* Response struct */
} NPF_F_ATMOamTxCallbackData_t;

/* Type for a callback function to be registered with ATM OAM Tx */
typedef void (*NPF_F_ATMOamTxCallbackFunc_t) (
    NPF_IN NPF_userContext_t userContext,
    NPF_IN NPF_correlator_t correlator,
    NPF_IN NPF_F_ATMOamTxCallbackData_t data);

/* Completion Callback Registration Function */
NPF_error_t NPF_F_ATMOamTxRegister (
    NPF_IN NPF_userContext_t userContext,
    NPF_IN NPF_F_ATMOamTxCallbackFunc_t callbackFunc,
    NPF_OUT NPF_callbackHandle_t *callbackHandle);

/* Completion Callback Deregistration Function */
NPF_error_t NPF_F_ATMOamTxDeregister (
    NPF_IN NPF_callbackHandle_t callbackHandle);

/* LFB Attributes Query Function */
NPF_error_t NPF_F_ATMOamTxLFB_AttributesQuery (
    NPF_IN NPF_callbackHandle_t callbackHandle,
    NPF_IN NPF_correlator_t correlator,
    NPF_IN NPF_errorReporting_t errorReporting,
    NPF_IN NPF_FE_Handle_t feHandle,
    NPF_IN NPF_BlockId_t blockId);

#ifdef __cplusplus
}
#endif

#endif /* __NPF_F_ATM_OAMTX_H__ */

```

Appendix B Acknowledgements

Working Group Chair: Alex Conta

Task Group Chair: Per Wollbrand

The following individuals are acknowledged for their participation to ATM Task Group 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:

Pål Damnvik, Ericsson
Patrik Herneld, Ericsson
Ajay Kamalvanshi, Nokia
Jaroslaw Kogut, Intel
Arthur Mackay, Freescale
Stephen Nadas, Ericsson
Michael Persson, Ericsson
John Renwick, Agere Systems
Vedvyas Shanbhogue (ed.), Intel
Keith Williamson, Motorola
Weislaw Wisniewski, Intel
Per Wollbrand, Ericsson

Appendix C **List of companies belonging to NPF During Approval Process**

| | | |
|-------------------------|--------------------------|-------------------|
| Agere Systems | IDT | Sensory Networks |
| AMCC | Infineon Technologies AG | Sun Microsystems |
| Analog Devices | Intel | Teja Technologies |
| Cypress Semiconductor | IP Fabrics | TranSwitch |
| Enigma Semiconductor | IP Infusion | U4EA Group |
| Ericsson | Motorola | Wintegra |
| Flextronics | Mercury Computer Systems | Xelerated |
| Freescale Semiconductor | Nokia | Xilinx |
| HCL Technologies | NTT Electronics | |
| Hifn | PMC-Sierra | |