ATM Traffic Manager LFB and Functional API Implementation Agreement

August 16, 2005
Revision 1.0

Editor:

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

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

List of Tables

1 Revision History

Revision	Date	Reason for Changes
1.0	08/16/2005	Rev 1.0 of the ATM Traffic Manager LFB and Functional API Implementation Agreement. Source: npf2004.156.12.

2 Introduction

This contribution defines the ATM Traffic Manager LFB and lists configurations that are required in the LFB.

2.1 Acronyms

- **ABR**: Available Bit Rate
- **ATM**: Asynchronous Transfer Mode
- **API**: Application Programming Interface
- **CLP**: Cell Loss Priority
- **CBR**: Constant Bit Rate
- **CDVT**: CDV Tolerance
- **FAPI**: Functional API
- **GFR**: Guaranteed Frame Rate
- **IA**: Implementation Agreement
- **ID**: Identifier
- **LFB**: Logical Functional Block
- **LP:** Loss Priority
- **MBS**: Maximum Burst Size
- **MCR**: Minimum Cell Rate
- **MFS**: Maximum Frame Size
- **Nrt-VBR**: Non-Real-time VBR
- **NNI**: Network Node Interface
- **PCR**: Peak Cell Rate
- **PTI**: Payload Type Indicator
- **SCR**: Sustainable Cell Rate
- **SDU**: Service Data Unit
- **TM**: Traffic Management
- **UBR**: Unspecified Bit Rate
- **UNI**: User Network Interface
- **UPC**: Usage Parameter Control
- **VBR**: Variable Bit Rate

2.2 Assumptions

The ATM Traffic Manager 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 traffic management on ATM virtual links (VP/VC links) to ensure compliance to the established traffic contract. 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 Traffic Manager Description

The ATM Traffic Manager is used to ensure conformance of the traffic on a virtual link to the established traffic contract for a virtual link by using mechanisms like buffering, traffic shaping, queuing and scheduling, etc. on the virtual links of the connection. The ATM Traffic Manager LFB is modeled as shown in Figure 3.1

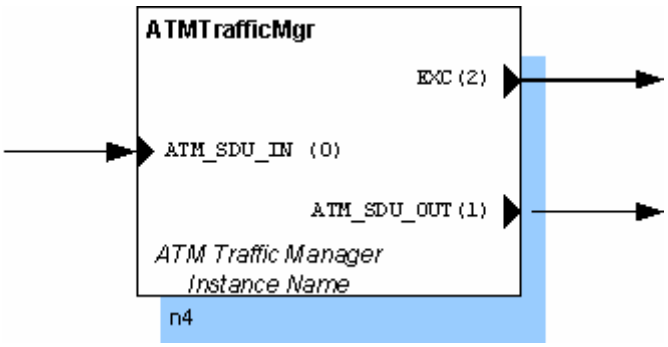

Figure 3.1: ATM Traffic Manager LFB

The LFB may contain multiple instances of traffic manager each associated with a VP or VC links. Such instances are depicted in Figure 3.2. Incoming ATM SDU are assigned to appropriate traffic manager instance according to metadata received with ATM SDU identifying the link on which the SDU is to be transmitted. More than one traffic manager instances may process an ATM SDU to perform hierarchical traffic management at VP and VC level. The maximum number of traffic managers is an attribute of the ATM Traffic Manager LFB and may be queried as such.

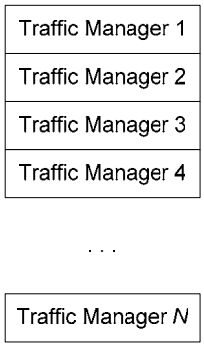

Figure 3.2: Traffic Manager Instances

3.1 ATM Traffic Manager Inputs

Table 3.1: ATM Traffic Manager LFB Inputs

Symbolic Name	Input ID	Description
ATM_SDU_IN	0	This is the only input for the ATM Traffic Manager LFB and is used to receive ATM SDU's to be transmitted on the virtual link specified by the metadata.

3.1.1 Metadata Required

Table 3.2: Input Metadata for ATM Traffic Manager 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/Re-write	Payload Type of ATM cell. May be modified if EFCI is changed.
META_ATM_LP	Read	Loss priority of the ATM cell.

3.2 ATM Traffic Manager Outputs

Table 3.3: ATM Traffic Manager LFB Outputs

Symbolic Name	Output ID	Description
ATM_SDU_OUT	1	This is the normal output for the ATM Traffic Manager LFB. ATM SDU's are sent on this output for transmission on the line.
EXC	2	The cell is sent to this output if the SDU has to be discarded due to various traffic management actions, lack of buffer space, etc.

3.2.1.1 Metadata Produced

Table 3.4: Output Metadata for ATM Traffic Manager LFB

Metadata tag	Access method	Description
META_ATM_PTI	Read/Re-write	Payload Type of ATM cell to be transmitted. May be modified if EFCI is changed.

3.3 Accepted Inputs

The ATM Traffic Manager LFB can accept any ATM SDU's for transmission over UNI or NNI.

3.4 Cell Modifications

The ATM SDU's received by the ATM Traffic Manager LFB are not subject to any modification and are not consumed by this LFB and always exit through one of the outputs. The ATM Traffic Manager LFB sequentially processes ATM SDU's entering the LFB input and belonging to a given virtual link. That means that ATM Traffic Manager LFB does not change the order of transmission of the ATM SDU's on a given virtual link.

3.5 Relationship with Other LFBs

The ATM Traffic Manager LFB is placed in the processing chain before the ATM Header Generator LFB. The ATM Traffic Manager LFB receives primarily ATM SDU's from previous LFB and passes them to the next LFB in chain after suitable traffic management actions like buffering, shaping, scheduling, etc. The sequence of actions that configures an ATM Traffic Manager LFB and cooperating AAL5 Transmit LFB instance, and cooperation between these two LFBs is schematically depicted in Figure 3.3.

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

The EXC output of the ATM Traffic Manager LFB could be connected to an LFB that receives SDU's which are the traffic manager considers not fit for transmission on the line due to reasons like lack of buffer space, etc. Depending on system design this may be either dropper, which drops SDU's or other LFB that makes a decision on how to utilize such SDU's.

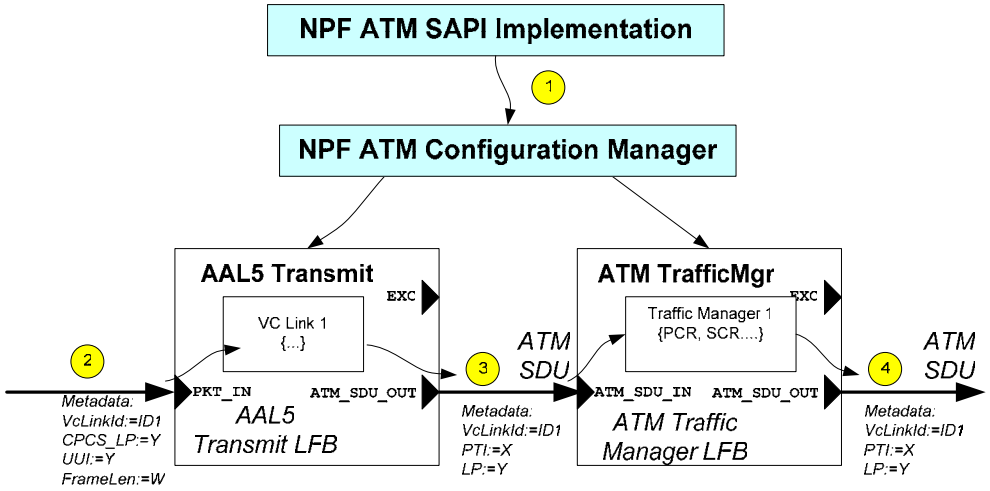

Figure 3.3: Cooperation between ATM Traffic Manager and AAL5 Transmit LFB

This figure shows part of example Forwarding Element that contains AAL5 Transmit LFB and ATM Traffic Manager LFBs. These two blocks are connected in chain and configured by the ATM configuration manager. The sequence of actions that configure a virtual channel link on the interface may be defined as follows (see corresponding numbers in circles in the figure):

1. The NPF ATM SAPI is invoked to create a terminated VC link. The system software under the NPF ATM SAPI assigns a VC Link ID 'ID1' to the VC link and invokes the ATM configuration manager FAPI to create the VC Link. This causes a VC link instance to be created in the AAL5

Transmit LFB. An instance of ATM Traffic Manager is created in the ATM Traffic Manager LFB to manage the traffic sent on the VC Link with VC Link ID ‘ID1’.

2. The AAL5 Transmit LFB receives a packet from the AAL5 service user LFB. The AAL5 Transmit LFB performs the segmentation of the packet in to ATM SDU’s.
3. The ATM SDU is forwarded to the ATM Traffic Manager LFB along with the PTI and LP over the `ATM_SDU_OUT` output of the AAL5 Transmit LFB. The ATM Traffic Manager LFB uses the VC Link ID from the metadata to determine the Traffic Manager instance associated with this VC link and performs the required traffic management actions like buffering, shaping, etc. as configured by the service category associated with the VC link. Additionally, the ATM traffic manager may carry out a another level of traffic management on the ATM SDU using the Traffic Manager instance configured for the VP Link carrying the VC Link with ID ‘ID1’.
4. When the ATM Traffic Manager determines it is time to schedule transmission of an ATM cell on that VC Link, the ATM SDU is forwarded to the next LFB in the chain for further processing.

.

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 Traffic Manager LFB in a forwarding element using a block type value for the ATM Traffic Manager LFB.

```
#define NPF_F_ATMTRAFFICMANAGER_LFB_TYPE  36
```

4.1.2 ATM Traffic Manager Configurations

The ATM Traffic Manager LFB requires below configurations for traffic manager configured for each virtual link.

- Virtual Link ID
- Virtual Link Type – VP or VC Link
- Service category of the connection - CBR, rt-VBR, nrt-VBR, ABR, UBR, GFR, UBR with PCR, UBR without PCR, UBR with minimum desired cell rate (MDCR), UBR with MDCR and PCR, other
- Peak cell rate (PCR)
- Sustainable cell rate (SCR)
- Maximum burst size (MBS)
- Minimum cell rate (MCR)
- Maximum frame size (MFS)
- Cell delay variation tolerance (CDVT)
- Priority associated with UBR service
- Buffer threshold configured for each virtual link
- Queueing cell drop policy

4.2 Data Structures for Completion Callbacks

4.2.1 ATM Traffic Manager LFB Attributes query response

The attributes of an ATM Traffic Manager LFB are the following:

```
typedef struct {
   NPF_uint32_t   maxTrafficManagers;     /* Maximum possible TMs  */
   NPF_uint32_t   curNumTrafficManagers;  /* Current number of TMs */
}  NPF_F_ATMTrafficManagerLFB_AttrQueryResponse_t;
```

The `maxTrafficManagers` field contains the maximum number of traffic managers supported in this ATM Traffic Manager LFB. The `curNumTrafficManagers` field contains the number of traffic managers currently established in the ATM Traffic Manager LFB.

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_ATMTrafficManagerErrorType_t error; /* Error code */
    union {
        /* NPF_F_ATMTrafficManagerLFB_AttributesQuery() */
```

```
        NPF_F_ATMTrafficManagerLFB_AttrQueryResponse_t LFB_AttrQueryResp;
    } u;
} NPF_F_ATMTrafficManagerAsyncResp_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_ATMTrafficManagerCallbackData_t.
*/
typedef enum NPF_F_ATMTrafficManagerCallbackType {
    NPF_F_ATMTRAFFICMANAGER_ATTR_QUERY = 1,
} NPF_F_ATMTrafficManagerCallbackType_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_ATMTrafficManagerCallbackData_t` structure.

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

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 Traffic Manager 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 Traffic Manager Configuration and management APIs error codes. These codes are used in callbacks to deliver results of the requested operations.

```
/* Asynchronous error codes (returned in function callbacks) */
typedef NPF_uint32_t NPF_F_ATMTrafficManagerErrorType_t;
#define NPF_ATMTRAFFICMANAGER_BASE_ERR\
                                    (NPF_F_ATMTRAFFICMANAGER_LFB_TYPE * 100)
#define ATMTRAFMGR_ERR(n) ((NPF_F_ATMTrafficManagerErrorType_t)\
           (NPF_ATMTRAFFICMANAGER_BASE_ERR+ (n))
#define NPF_E_ATMTRAFMGR_INVALID_TRAFMGR_BLOCK_ID ATMTRAFMGR_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_ATMTrafficManagerCallbackFunc_t) (
      NPF_IN NPF_userContext_t            userContext,
      NPF_IN NPF_correlator_t             correlator,
      NPF_IN NPF_F_ATMTrafficManagerCallbackData_t    data);
```

5.2.1.1 Description

This callback function is for the application to register an asynchronous response handling routine to the ATM Traffic Manager API implementation. This callback function is intended to be implemented by the application, and be registered to the ATM Traffic Manager API implementation through the `NPF_F_ATMTrafficManagerRegister` 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 Traffic Manager 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_ATMTrafficManagerRegister(
      NPF_IN NPF_userContext_t                   userContext,
      NPF_IN NPF_F_ATMTrafficManagerCallbackFunc_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 Traffic Manager 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 Traffic Manager 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 Traffic Manager 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_ATMTrafficManagerDeregister(
      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 Traffic Manager 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_ATMTrafficManagerLFB_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 Traffic Manager LFB's attributes at a time. If the ATM Traffic Manager 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 Traffic Manager 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 Traffic Manager 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_ATMTRAFMGR_INVALID_TRAFMGR_BLOCK_ID` – LFB ID is not an ID of LFB that has ATM Traffic Manager functionality

The `LFB_AttrQueryResponse` field of the union in the `NPF_F_ATMTrafficManagerAsyncResponse_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_IA.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 Traffic Manager Functional API
 */

#ifndef __NPF_F_ATM_TRAFFICMANAGER_H__
#define __NPF_F_ATM_TRAFFICMANAGER_H__

#ifdef __cplusplus
extern "C" {
#endif

/* It is possible to use the FAPI Topology Discovery
   APIs to discover an ATM Traffic Manager LFB
   in a forwarding element.  */
#define NPF_F_ATMTRAFFICMGR_LFB_TYPE      36

/* Asynchronous error codes (returned in function callbacks) */
typedef NPF_uint32_t NPF_F_ATMTrafficManagerErrorType_t;
#define NPF_ATMTRAFFICMANAGER_BASE_ERR\
                                    (NPF_F_ATMTRAFFICMANAGER_LFB_TYPE * 100)

#define ATMTRAFMGR_ERR(n) ((NPF_F_ATMTrafficManagerErrorType_t)\
            (NPF_ATMTRAFFICMANAGER_BASE_ERR+ (n))

#define NPF_E_ATMTRAFMGR_INVALID_TRAFMGR_BLOCK_ID ATMTRAFMGR_ERR(0)

/*************************************************************
 * Enumerations and types for ATM Traffic Manager attributes and*
 * completion callback data types                              *
 *************************************************************/

/* The attributes of an ATM Traffic Manager */
typedef struct {
      NPF_uint32_t   maxTrafficManagers;     /* Maximum possible TMs  */
      NPF_uint32_t   curNumTrafficManagers;  /* Current number of TMs */
}  NPF_F_ATMTrafficManagerLFB_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_ATMTrafficManagerErrorType_t error; /* Error code */
    union {
        /* NPF_F_ATMTrafficManagerLFB_AttributesQuery() */
        NPF_F_ATMTrafficManagerLFB_AttrQueryResponse_t LFB_AttrQueryResp;
    } u;
} NPF_F_ATMTrafficManagerAsyncResp_t;

/*
* Completion Callback Types, to be found in the callback
* data structure, NPF_F_ATMTrafficManagerCallbackData_t.
*/
typedef enum NPF_F_ATMTrafficManagerCallbackType {
    NPF_F_ATMTRAFFICMANAGER_ATTR_QUERY = 1,       /* Attributes Query   */
} NPF_F_ATMTrafficManagerCallbackType_t;
```

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

/* Type for a callback function to be registered with ATM traffic Mgr */
typedef void (*NPF_F_ATMTrafficManagerCallbackFunc_t) (
      NPF_IN NPF_userContext_t            userContext,
      NPF_IN NPF_correlator_t             correlator,
      NPF_IN NPF_F_ATMTrafficManagerCallbackData_t    data);


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

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

/* LFB Attributes Query Function */
NPF_error_t NPF_F_ATMTrafficManagerLFB_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_TRAFFICMANAGER_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 Dammvik, 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
AMCC
Analog Devices
Cypress Semiconductor
Enigma Semiconductor
Ericsson
Flextronics
Freescale Semiconductor
HCL Technologies
Hifn
IDT
Infineon Technologies AG
Intel
IP Fabrics
IP Infusion
Motorola
Mercury Computer Systems
Nokia
NTT Electronics
PMC-Sierra
Sensory Networks
Sun Microsystems
Teja Technologies
TranSwitch
U4EA Group
Wintegra
Xelerated
Xilinx