# Switch Address Generator ATM LFB and Functional API Implementation Agreement

May 29, 2006
Revision 1.0

**Editor: Patrik Herneld, Ericsson,** patrik.herneld@ericsson.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

## List of Tables

# Glossary

Table 1 Glossary

| AAL | ATM Adaptation Layer: The standards layer that allows multiple applications to have data converted to and from the ATM cell. A protocol used that translates higher layer services into the size and format of an ATM cell.<br>Examples of AALs are AAL1, AAL2 and AAL5. |
|---|---|
| AAL2 | ATM Adaptation Layer 2 defined in ITU-T I.363.2. The type of ATM adaptation principally used for variable-bit-rate Voice-Over-ATM services. |
| AAL5 | ATM Adaptation Layer 5 defined in ITU-T I.363.5. The type of ATM adaptation principally used for frame and packet transport over an ATM network. |
| ATM | Asynchronous Transfer Mode |
| BP | Backplane |
| CLP | Cell Loss Priority |
| CPCS | Common Part Convergence Sublayer |
| CPS | Common Part Sub-Layer |
| HEC | Header Error Control |
| IA | Implementation Agreement |
| ID | Identifier |
| LI | Length Indication |
| NNI | Network Node Interface |
| PDU | Protocol Data Unit |
| PTI | Payload Type Identifier |
| SAR | Segmentation and Reassembly (Sublayer) |
| SDU | Service Data Unit |
| SSSAR | Service Specific Segmentation and Reassembly sublayer |
| UNI | User Network Interface |
| UUI | User-to-User Indication |
| VC | Virtual Connection |
| VCC | Virtual Channel Connection |
| VCI | Virtual Channel Identifier |
| VPC | Virtual Path Connection |
| VPI | Virtual Path Identifier |

# 1    Revision History

Table 2 Revision History

| Revision | Date | Reason for Changes |
|----------|------|--------------------|
| 1.0 | 05/29/2006 | Rev 1.0 of the Switch Address Generator ATM LFB and Functional API Implementation Agreement. Source : npf2005.048.11 |

# 2  Introduction

This contribution defines the Switch Address Generator ATM and lists configurations that are required in the LFB.

## 2.1  *Purpose and motivation*

The purpose of this LFB is to generate a backplane PDU, which is to be forwarded over a backplane between FEs, which could be used and formatted by a following Messaging LFB. A header or trailer is added to the incoming packet (e.g. ATM SDU), the result is called the backplane PDU.

### 2.1.1  Optional formatting function

The Messaging LFB can alter the location of a metadata within the metadata stream and reduce the size of a metadata to a minimum of a byte granule.

Many implementations would like to construct e.g. an internal ATM Header, as part of the so-called switch address (configured through the ATM Configuration Manager Functional API). This would require a merge of incoming metadata and generated metadata on a bit level. This merge cannot be done with the Messaging LFB (without a FAPI implementation change for each wanted bit merging – not very flexible solution).

Furthermore when e.g. adding and deleting ATM VCs that are cross connected over different backplane links, a switch address needs to be configured for each VC link through the ATM Configuration Manager Functional API. This switch address will be distributed to the Switch Address Generator ATM LFB, and will be used to generate the backplane PDU. In many implementations there are octets in the switch address that are static and common for all backplane links within the LFB. An example of such fields could be the HEC field, which might not be generated per VC inside a switch and would therefore be the same for all VCs.

To optimize performance it would be useful to be able to configure the LFB once with the static octets that are to be generated for each backplane link instance within the LFB. These octets can then be omitted in the frequent configurations of e.g. ATM VCs within the ATM Configuration Manager Functional API.

Therefore the Switch Address Generator ATM contains the optional function, `NPF_F_SwAddrGenATM_Format()`. This function can merge incoming metadata or static defined strings into the txAddress (which forms the header or trailer part of the backplane PDU). The optional function can also specify the placement of the metadata-generated part in relation to the packet (header or trailer).

## 2.2  *Assumptions*

The Switch Address Generator ATM LFB obtains its configurations from the ATM Configuration Manager LFB. 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 backplane links. The IA also specifies the metadata generated and consumed by this LFB.

## 2.4  *External Requirements and Dependencies*

This IA depends on the following IA's:

- o  NPF Software API Conventions Implementation Agreement document [SWAPICON] for basic type definitions.
- o  ATM Software API Architecture Framework defines the architectural framework for the ATM FAPIs

o ATM Configuration Manager Functional API defines the functions for configuration and management of the ATM LFBs on a forwarding element.

o This document depends on Software API Conventions Implementation agreement Revision 2.0 for below type definitions

  o NPF_error_t – Refer section 5.2 of Software API Conventions IA Rev 2.0

  o NPF_callbackHandle_t - Refer section 5.2 of Software API Conventions IA Rev 2.0

  o NPF_userContext_t - Refer section 5.2 of Software API Conventions IA Rev 2.0

  o NPF_errorReporting_t - Refer section 5.2 of Software API Conventions IA Rev 2.0

  o NPF_boolean_t - Refer section 5.1 of Software API Conventions IA Rev 2.0

  o NPF_uint8_t - Refer section 5.1 of Software API Conventions IA Rev 2.0

  o NPF_uint32_t - Refer section 5.1 of Software API Conventions IA Rev 2.0

  o NPF_correlator_t - Refer section 5.2 of Software API Conventions IA Rev 2.0

o This document depends on Topology Manager Functional API Implementation Agreement Revision 1.0 for below type definitions

  o NPF_BlockId_t – Refer section 3.1.1 of Topology Manager Functional API IA Rev 1.0

  o NPF_FE_Handle_t – Refer section 3.1.1 of Topology Manager Functional API IA Rev 1.0

# 3    Switch Address Generator ATM LFB Description

The Switch Address Generator ATM LFB receives a packet on one of the four inputs from the previous LFB in the pipeline, performs a lookup to find the backplane link, produces the required backplane PDU and forwards the PDUs to the next LFB. The Switch Address Generator ATM LFB uses the BP Link ID signaled in the metadata to find the backplane link to be used to transport the packet over the backplane.

The Switch Address Generator ATM LFB is modeled as shown in Figure 1:
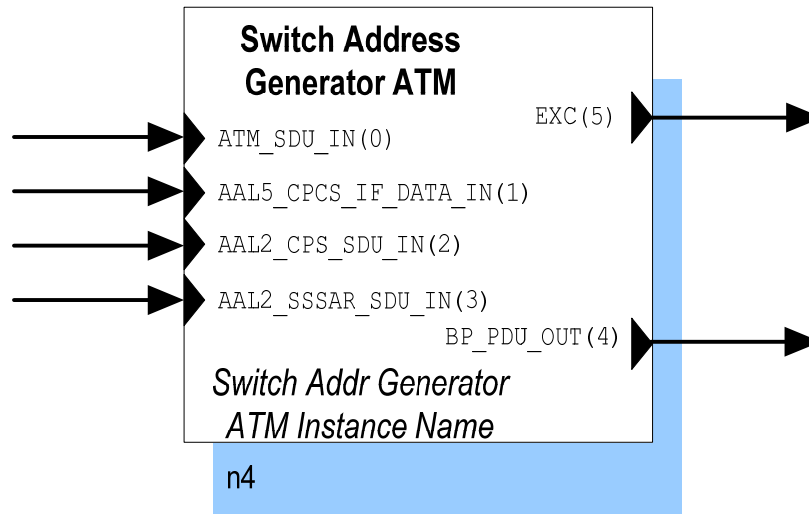


Figure 1 Switch Address Generator ATM LFB

The Switch Address Generator ATM LFB may contain multiple instances of backplane links that are identified by unique backplane link IDs. The incoming packets are associated with the appropriate backplane link instance using the metadata received with the packet. Such backplane link instances are depicted in **Figure 2** below. The maximum number of such backplane links is an attribute of the Switch Address Generator ATM LFB and may be queried as such.
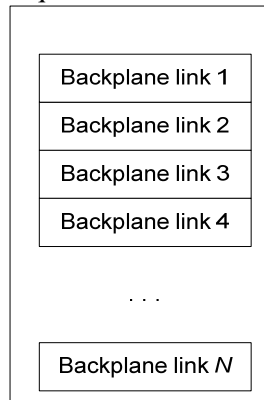


Figure 2 Backplane link Instances

The Switch Address Generator ATM LFB maintains the following statistics for each backplane link:

- None.

The Switch Address Generator ATM LFB forms the output backplane PDU as shown, by using an ATM SDU as example, in **Figure 3** below. The metadata-generated portion of the PDU can be placed either in front of the ATM SDU as a header (default) or as a trailer to the ATM SDU. The metadata-generated portion of the backplane PDU is based on the txAddress (from the SwitchAddress within the ATM Configuration Manager); an optional function (NPF_F_SwAddrGenATM_Format ()) could be used to merge incoming metadata or static defined strings into the txAddress. The optional function can also specify the placement of the metadata-generated part in relation to the ATM SDU (header or trailer). All four inputs of the Switch Address Generator ATM LFB form the backplane PDU in the same way as shown in this example.
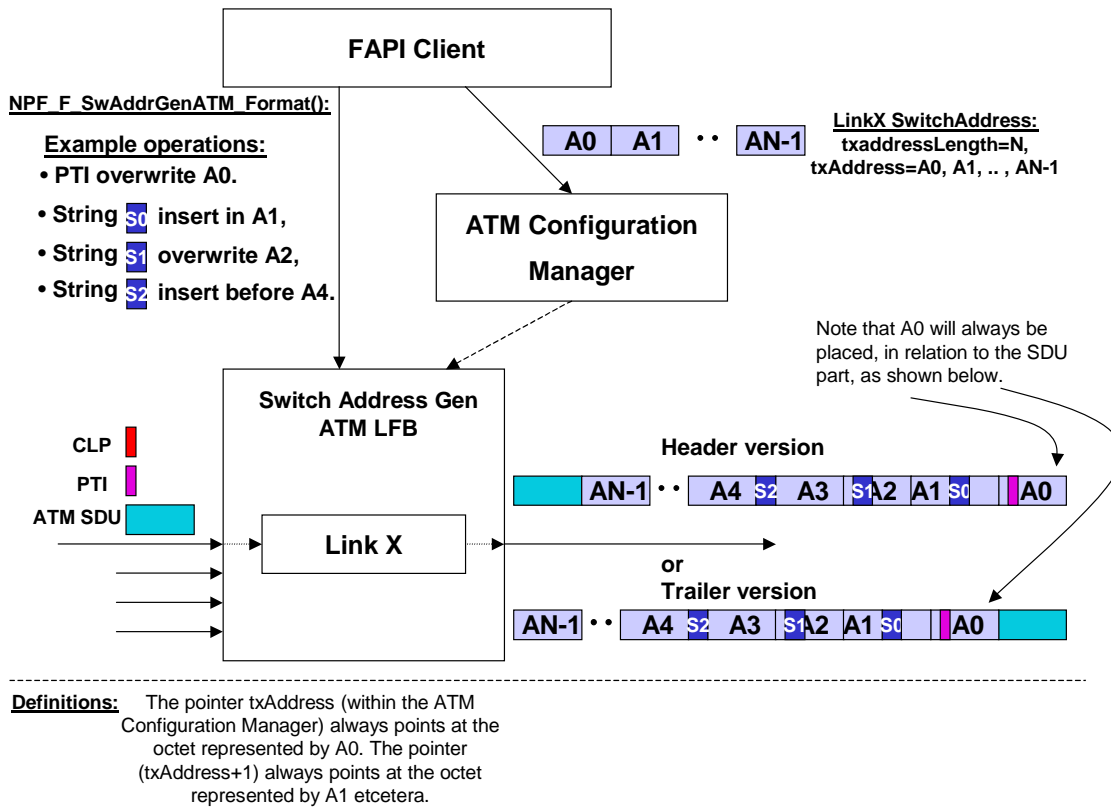


Figure 3 Overview of the LFB functionality

## 3.1 Switch Address Generator ATM Inputs

Table 3 Switch Address Generator ATM LFB Inputs

| Symbolic Name | Input ID | Description |
|---|---|---|
| ATM_SDU_IN | 0 | This is the ATM SDU input for the Switch Address Generator ATM LFB that is used to receive ATM SDUs to be sent over the backplane link. |
| AAL5_CPCS_IF_DATA_IN | 1 | This is the AAL5 packet input for the Switch Address Generator ATM LFB that is used to receive AAL5 packets to be sent over the backplane link. |
| AAL2_CPS_SDU_IN | 2 | This is the AAL2 CPS SDU input for the Switch Address Generator ATM LFB that is used to receive AAL2 CPS SDUs to be sent over the backplane link. |
| AAL2_SSSAR_SDU_IN | 3 | This is the AAL2 SSSAR SDU input for the Switch Address Generator ATM LFB that is used to receive AAL2 SSSAR SDUs to be sent over the backplane link. |

## 3.1.1 Metadata Required

Table 4 Input Metadata for Switch Address Generator ATM LFB

| Metadata tag | Access method | Description |
|---|---|---|
| META_BPL_ID | Read-and-consumed | Metadata identifying the Backplane link on which the Backplane PDU is to be transmitted. This metadata is received on all four inputs respectively. |
| META_ATM_PTI | Read | Conditional metadata representing the Payload Type of the ATM SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the ATM_SDU_IN input only. |
| META_ATM_LP | Read | Conditional metadata representing the Loss Priority of the ATM SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the ATM_SDU_IN input only. |
| META_ATM_VCI | Read | Conditional metadata representing the VCI of the ATM SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the ATM_SDU_IN input only. |
| META_AAL5_CPCS_LP | Read | Conditional metadata representing the CPCS Loss Priority of the re-assembled AAL5 packet. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL5_CPCS_IF_DATA_IN input only. |
| META_AAL5_UUI | Read | Conditional metadata representing the CPCS UUI of the re-assembled AAL5 packet. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL5_CPCS_IF_DATA_IN input only. |
| META_AAL5_FRAME_LEN | Read | Conditional metadata representing the length of the re-assembled AAL5 packet. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL5_CPCS_IF_DATA_IN input only. |
| META_AAL5_BOUND_IF | Read | Conditional metadata representing the handle of the child interface bound to the link. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. |

| | | |
|---|---|---|
| | | This metadata is received on the AAL5_CPCS_IF_DATA_IN input only. |
| META_AAL5_RCV_STATUS | Read | Conditional metadata representing the type of error encountered in the reassembly process. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL5_CPCS_IF_DATA_IN input only. |
| META_AAL2_CPS_UUI | Read | Conditional metadata representing the UUI of the CPS packet. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_CPS_SDU_IN input only. |
| META_AAL2_CPS_LI | Read | Conditional metadata representing the length of the CPS SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_CPS_SDU_IN input only. |
| META_AAL2_CPS_SSCS_TYPE | Read | Conditional metadata representing the type of SSCS to process this CPS SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_CPS_SDU_IN input only. |
| META_AAL2_CPS_BOUND_IF | Read | Conditional metadata representing the handle of the higher layer interface bound to this AAL2 channel. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_CPS_SDU_IN input only. |
| META_AAL2_SSSAR_UUI | Read | Conditional metadata representing the UUI signaled in the re-assembled AAL2 SAR SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_SSSAR_SDU_IN input only. |
| META_AAL2_SSSAR_LI | Read | Conditional metadata representing the length of the re-assembled AAL2 SAR SDU. This metadata only needs to be read in case the optional function is being used and the function needs this metadata. This metadata is received on the AAL2_SSSAR_SDU_IN input only. |

## *3.2 Switch Address Generator ATM Outputs*

Table 5 Switch Address Generator ATM LFB Outputs

| Symbolic Name | Output ID | Description |
|---|---|---|
| BP_PDU_OUT | 4 | This is the normal output for the Switch Address Generator ATM LFB. Backplane PDUs that could be identified with an established backplane link are sent on this output for further processing. Adding a header or a trailer part in front/behind the input packet generates the backplane PDUs. The header/trailer is generated from the txAddress (ATM Configuration Manager) that may be formatted by an optional function (within the Switch Address Generator ATM LFB). |
| EXC | 5 | The cell is sent to this output if an error is encountered. |

## **3.2.1 Metadata Produced**

Table 6 Output Metadata for Switch Address Generator ATM LFB

| Metadata tag | Access method | Description |
|---|---|---|
| META_BP_IF_ID | Write | Metadata to associate a backplane interface on which the Backplane PDU is to be transmitted over. |
| META_BP_PDU_TYPE | Write | Metadata to associate what type the backplane PDU consists of. This metadata links the transmitted PDU to one of the four inputs of the Switch Address Generator ATM LFB; it could be used by e.g. the Messaging LFB to associate a format profile to the PDU. |

## *3.3 Accepted Packet Types*

The Switch Address Generator ATM LFB can accept ATM SDUs, AAL5 packets, AAL2 CPS SDUs and AAL2 SSSAR SDUs received on its inputs.

## *3.4 Packet Modifications*

The different packets received by the Switch Address Generator ATM LFB are not consumed or modified by the LFB and always exit through one of the outputs. The Switch Address Generator ATM LFB processes packets entering each one of the LFB inputs sequentially. That means that the Switch Address Generator ATM LFB does not change the order of transmission of the packets.

## *3.5 Relationship with Other LFBs*

The Switch Address Generator ATM LFB could e.g. be placed in the processing chain before the Messaging LFB. The Switch Address Generator ATM LFB receives packets from the previous LFB and passes the Backplane PDUs to the Messaging LFB for transmission over a backplane.

The recipient and producers of the SDUs and metadata that are described in this section may be replaced by LFBs that are able to generate information that is required by the Switch Address Generator ATM LFB at its input, and are able to utilize information present at the output of the Switch Address Generator ATM LFB. The exact design and connections between the Switch Address Generator ATM LFB and cooperating blocks is specific to the vendor that provides the Forwarding Element design and FAPI implementation.

The EXC output of the Switch Address Generator ATM LFB could be connected to an LFB that receives SDUs for which the proper backplane link instance could not be found. Depending on system design this may be either a dropper, which drops SDUs that could not be properly associated with a backplane link, or other LFB that makes a decision how to utilize such SDUs.

The sequence of actions that configures the Switch Address Generator ATM LFB and cooperating Messaging LFB instance, and cooperation between these two LFBs is schematically depicted in **Figure 4**. Note that only the ATM_SDU_IN input is shown in the figure as an example.
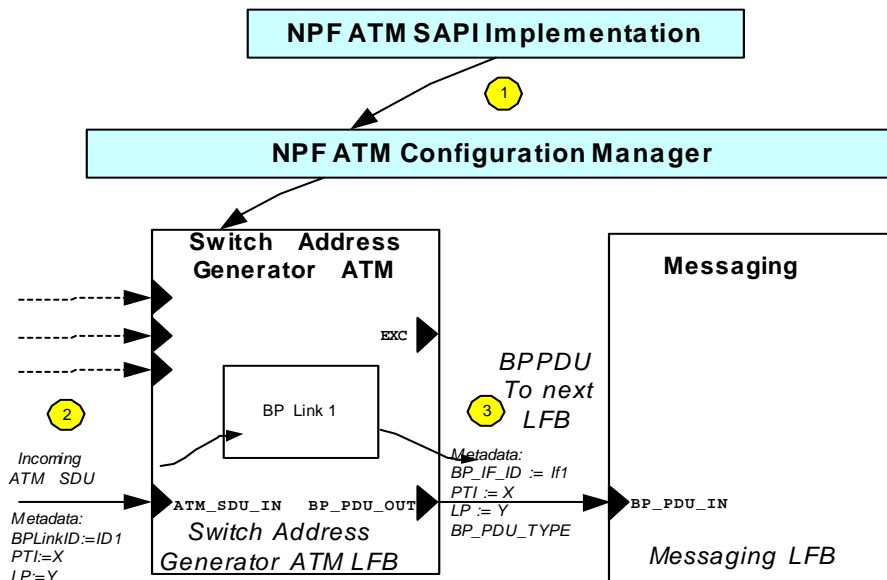


Figure 4 Cooperation between Switch Address Generator ATM LFB and Messaging LFB

This figure shows part of an example Forwarding Element that contains the Switch Address Generator ATM and the Messaging LFBs. These two blocks are connected in chain and configured by a NPF SAPI implementation. The sequence of actions that configure the backplane link on the interface may be defined as follows (see corresponding numbers in circles in the figure):

1.  The NPF ATM SAPI Implementation creates a backplane link instance. The system software below the NPF ATM SAPI assigns a BP Link ID ('ID1') to the backplane link and invokes the

ATM configuration manager FAPI to create the link. The ATM configuration manager FAPI call leads to creation of a backplane link instance in the Switch Address Generator ATM LFB.

2. The Switch Address Generator ATM LFB uses the backplane Link ID signaled in metadata to find a matching backplane link instance on receiving an ATM SDU. The PTI and LP signaled in the metadata are passed transparent to the Switch Address Generator ATM LFB.

3. The generated Backplane PDU is forwarded to the Messaging LFB input together with metadata created by the Switch Address Generator ATM LFB.

# 4    Data Types

## *4.1    Common LFB Data Types*

### 4.1.1  LFB Type Code

It is possible to use the FAPI Topology Manager APIs to discover a Switch Address Generator ATM LFB in a forwarding element using a block type value for the Switch Address Generator ATM LFB.

```
#define NPF_F_SWADDRGENATM_LFB_TYPE    45
```

### 4.1.2  Switch Address Generator ATM Configurations

#### 4.1.2.1 Backplane Link Characteristics

The Switch Address Generator LFB requires below configurations for each backplane link.

- Backplane Link ID
- TX Backplane Interface ID
- TX Address Length
- TX Address

### 4.1.3  Input Type

This type defines the LFB input to apply the formatting on.

```
typedef enum {
   NPF_F_SWADDRGENATM_ATM_SDU_IN = 0,           /* ATM SDU input. */
   NPF_F_SWADDRGENATM_AAL5_CPCS_IF_DATA_IN = 1, /* AAL5 packet input. */
   NPF_F_SWADDRGENATM_AAL2_CPS_SDU_IN = 2,      /* AAL2 CPS SDU input. */
   NPF_F_SWADDRGENATM_AAL2_SSSAR_SDU_IN = 3     /* AAL2 SSSAR SDU input. */
} NPF_F_SwAddrGenATM_Input_t;
```

### 4.1.4  ATM SDU metadata Type

This type defines the metadata on the ATM_SDU_IN input that can be formatted and put into the output header/trailer part of the Backplane PDU.

```
typedef enum {
   NPF_F_SWADDRGENATM_ATM_SDU_PTI = 0,  /* Payload Type Identifier */
   NPF_F_SWADDRGENATM_ATM_SDU_LP = 1,   /* Loss Priority */
   NPF_F_SWADDRGENATM_ATM_SDU_VCI = 2   /* Virtual Channel Identifier */
} NPF_F_SwAddrGenATM_AtmSduMetadata_t;
```

### 4.1.5  AAL5 packet metadata Type

This type defines the metadata on the AAL5_CPCS_IF_DATA_IN input that can be formatted and put into the output header/trailer part of the Backplane PDU.

```
typedef enum {
   NPF_F_SWADDRGENATM_AAL5_CPCS_LP = 0, /* CPCS Loss Priority */
   NPF_F_SWADDRGENATM_AAL5_UUI = 1,     /* CPCS UUI */
   NPF_F_SWADDRGENATM_AAL5_FRAMELENGTH = 2, /* Length of the packet */
   NPF_F_SWADDRGENATM_AAL5_BOUND_IF = 3,/* Handle of the child interface */
   NPF_F_SWADDRGENATM_AAL5_RCV_STATUS = 4 /* Type of error in reassembly */
} NPF_F_SwAddrGenATM_Aal5CpcsMetadata_t;
```

## 4.1.6 AAL2 CPS SDU metadata Type

This type defines the metadata on the AAL2_CPS_SDU_IN input that can be formatted and put into the output header/trailer part of the Backplane PDU.

```
typedef enum {
   NPF_F_SWADDRGENATM_AAL2_CPS_UUI = 0, /* CPS UUI */
   NPF_F_SWADDRGENATM_AAL2_CPS_LI = 1,  /* CPS LI */
   NPF_F_SWADDRGENATM_AAL2_BOUND_IF = 2 /* Handle of higher layer i/f */
} NPF_F_SwAddrGenATM_Aal2CpsSduMetadata_t;
```

## 4.1.7 AAL2 SSSAR SDU metadata Type

This type defines the metadata on the AAL2_SSSAR_SDU_IN input that can be formatted and put into the output header/trailer part of the Backplane PDU.

```
typedef enum {
   NPF_F_SWADDRGENATM_AAL2_SSSAR_UUI = 0,     /* UUI of SAR SDU */
   NPF_F_SWADDRGENATM_AAL2_SSSAR_LI = 1 /* LI of SAR SDU */
} NPF_F_SwAddrGenATM_Aal2SssarSduMetadata_t;
```

## 4.1.8 String Type

This type defines the string that can be formatted and put into the output header/trailer part of the Backplane PDU.

```
typedef struct {
   NPF_uint32_t stringLength;     /* Length of the String. */
   /* Pointer to an array of string octets. */
   NPF_uint8_t      *string;
} NPF_F_SwAddrGenATM_String_t;
```

## 4.1.9 Format Type

This type defines the metadata format structure.

```
typedef struct {
   /* Specifies which of the four inputs to apply the formatting to. */
   NPF_F_SwAddrGenATM_Input_t           inputToFormat;
   /* Specifies if the formatting operation shall use an input metadata
    * or a static string. If set to NPF_TRUE the 'metadataIn' is used,
    * if set to NPF_FALSE 'string' is used. */
   NPF_boolean_t                   metadata;
   /* Specifies if the metadata/string shall be overwriting existing bits
    * of the txAddress or be inserted at a specified bit position. If
    * set to NPF_TRUE an overwrite operation is performed, if set to
    * NPF_FALSE an insert operation is performed. */
   NPF_boolean_t                   overwrite;
   union {
     /* The metadata input to be used for the formatting. */
     union {
        NPF_F_SwAddrGenATM_AtmSduMetadata_t atmSduMetaIn;
        NPF_F_SwAddrGenATM_Aal5CpcsMetadata_t aal5CpcsMetaIn;
        NPF_F_SwAddrGenATM_Aal2CpsSduMetadata_t aal2CpsSduMetaIn;
        NPF_F_SwAddrGenATM_Aal2SssarSduMetadata_t aal2SssarSduMetaIn;
     }u1;
     /* The string to be used for the formatting. */
      NPF_F_SwAddrGenATM_String_t  string;
```

```
    }u2;
    /* The start bit of where the input metadata/string shall be overwriting
     * existing bits or be inserted. See Figure 5 and Figure 6 for an
     * overview of the bit numbering definition. */
    NPF_uint32_t                    startBit;
    /* The bit length of the input metadata/string that shall be overwriting
     * existing bits or be inserted. See Figure 5 and Figure 6 for an
     * overview of the bit numbering definition. For an insert operation
     * the bitlength MUST be a multiple of 8 (i.e. 8, 16, ..). An overwrite
     * operation has no such limitations. */
    NPF_uint32_t                    bitLength;
    /* This parameter specifies the order of formatting. If numEntries = 1,
     * the formatOrder MUST be set to '1'. If numEntries > 1, the formatOrder
     * must be set to '1, 2, 3, ... , numEntries' respectively. The format
     * operation with formatOrder = 1 SHALL always be executed first, the
     * format operation with formatOrder = 2 SHALL always be executed second
     * and so on. */
    NPF_uint32_t                    formatOrder;
} NPF_F_SwAddrGenATM_Format_t;
```
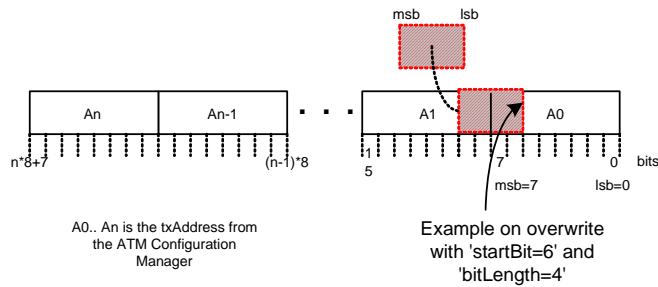
Figure 5 Bit numbering, showing an overwrite operation
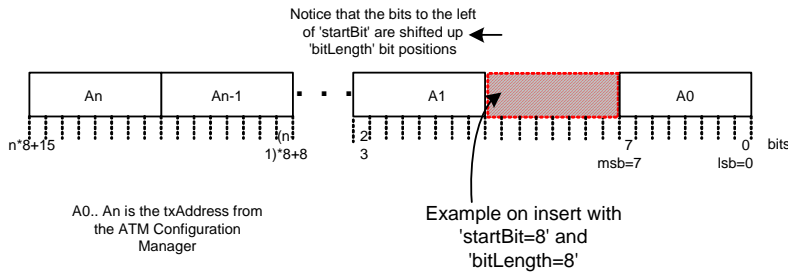
Figure 6 Bit numbering, showing an insert operation

## *4.2    Data Structures for Completion Callbacks*

### 4.2.1  Switch Address Generator ATM LFB Attributes query response

The attributes of a Switch Address Generator ATM LFB are the following:

```
typedef struct {
    NPF_uint32_t maxBpls;          /* Maximum possible BP links   */
    NPF_uint32_t curNumBpls;       /* Current number of BP links  */
    NPF_uint32_t maxTxAddrLength;  /* Maximum possible TX Address length
                                    * that can be set by the Configuration
                                    * Manager. */
} NPF_F_SwAddrGenATM_LFB_AttrQueryResponse_t;
```

The `maxBpls` field contains the maximum number of backplane links supported in this Switch Address Generator ATM LFB. The `curNumBpls` field contains the current number of Backplane links configured in the Switch Address Generator ATM LFB. The `maxTxAddrLength` field contains the maximum TX Address length (used in the Configuration Manager) supported by this Switch Address Generator ATM 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_SwAddrGenATM_ErrorType_t error;/* Error code for response */
    union {
        /* NPF_F_SwAddrGenATM_LFB_AttributesQuery() */
        NPF_F_SwAddrGenATM_LFB_AttrQueryResponse_t  lfbAttrQueryResponse;
    } u;
} NPF_F_SwAddrGenATM_AsyncResp_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_SwAddrGenATM_CallbackData_t.
 */
typedef enum NPF_F_SwAddrGenATM_CallbackType {
    NPF_F_SWADDRGENATM_ATTR_QUERY = 1,
} NPF_F_SwAddrGenATM_CallbackType_t;
```

### 4.2.4  Callback Data

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

```
/*
 * The callback function receives the following.
 * For the completed request, the error code is specified in the
 * NPF_ATM_AsyncResponse_t structure, along with any other information
 */
typedef struct {
    NPF_F_SwAddrGenATM_CallbackType_t type; /* Which function called?   */
```

```
    NPF_IN NPF_BlockId_t blockId;/* ID of LFB generating callback */
    NPF_F_SwAddrGenATM_AsyncResp_t resp;/* response structure */
} NPF_F_SwAddrGenATM_CallbackData_t;
```

The callback data returned for different callback types is summarized in Table 7.


Table 7 Callback type to Callback data mapping table

| Callback Type | Callback Data |
|---|---|
| NPF_F_SWADDRGENATM_ATTR_QUERY | NPF_F_SwAddrGenATM_LFB_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 the Switch Address Generator ATM 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 the Switch Address Generator ATM 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_SwAddrGenATM_ErrorType_t;

#define NPF_SWADDRGENATM_BASE_ERR (NPF_F_SWADDRGENATM_LFB_TYPE * 100)
#define NPF_E_SWADDRGENATM_INVALID_BLOCK_ID
                                    (NPF_SWADDRGENATM_BASE_ERR + 0)
```

# 5    Functional APIs (FAPIs)

## *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

This callback function is for the application to register an asynchronous response handling routine to the Switch Address Generator ATM API implementation. This callback function is intended to be implemented by the application, and be registered to the Switch Address Generator ATM API implementation through the `NPF_F_SwAddrGenATM_Register` function.

### *Syntax*

```
typedef void (*NPF_F_SwAddrGenATM_CallbackFunc_t) (
    NPF_IN NPF_userContext_t                  userContext,
    NPF_IN NPF_correlator_t                   correlator,
    NPF_IN NPF_F_SwAddrGenATM_CallbackData_t  data);
```

### *Description*

This function is a routine to handle the Switch Address Generator ATM asynchronous responses.

### *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 Switch Address Generator ATM API function call was invoked.
- `data` - The response information related to the particular callback type.

### *Output Parameters*

None

### *Return Values*

None

### 5.2.2  Completion Callback Registration Function

### *Syntax*

```
NPF_error_t NPF_F_SwAddrGenATM_Register(
    NPF_IN NPF_userContext_t                  userContext,
    NPF_IN NPF_F_SwAddrGenATM_CallbackFunc_t  callbackFunc,
    NPF_OUT NPF_callbackHandle_t              *callbackHandle);
```

### *Description*

This function is used by an application to register its completion callback function for receiving asynchronous responses related to the Switch Address Generator ATM 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`.

## *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.

## *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 Switch Address Generator ATM API functions. It will also be used when deregistering the userContext and callbackFunc pair.

## *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.

## *Notes*

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

### 5.2.3 Completion Callback Deregistration Function

## *Syntax*

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

## *Description*

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

## *Input Parameters*

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

## *Output Parameters*

None

## *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.

## *Notes*

- This API function MAY be invoked by any application no longer interested in receiving asynchronous responses for the Switch Address Generator ATM 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

### *Syntax*

```
NPF_error_t NPF_F_SwAddrGenATM_LFB_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);
```

### *Description*

This function call is used to query ONLY one Switch Address Generator ATM LFBs attributes at a time. If the Switch Address Generator ATM LFB exists, the various attributes of this LFB are returned in the completion callback.

### *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 Switch Address Generator ATM LFB.

### *Output Parameters*

None

### *Return Values*

- `NPF_NO_ERROR` - The operation is in progress.
- `NPF_E_UNKNOWN` - The LFB attributes was not queried due to invalid Switch Address Generator ATM 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.

### *Asynchronous Response*

There may be multiple asynchronous callbacks to this request. Possible error codes are:
- `NPF_NO_ERROR` – Operation completed successfully.
- `NPF_E_SWADDRGENATM_INVALID_BLOCK_ID` – LFB ID is not an ID of LFB that has Switch Address Generator ATM functionality.

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

## 5.3.2  Formatting Function

### *Syntax*

```
NPF_error_t NPF_F_SwAddrGenATM_Format(
    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,
    NPF_IN NPF_boolean_t                    formatEnable,
    NPF_IN NPF_boolean_t                    header,
    NPF_IN NPF_uint32_t                     numEntries,
    NPF_IN NPF_F_SwAddrGenATM_Format_t      *formatArray);
```

### *Description*

This function call is used to overwrite or insert a static string or a metadata into the header/trailer part of the Backplane PDU. It is also possible to specify if the added octets to the SDU (which together forms the Backplane PDU) shall be placed as a header or a trailer. If this optional function is not being used, the default behavior of the LFB is to place the txAddress (from the ATM Configuration Manager) as a header with no formatting applied.

### *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 Switch Address Generator ATM LFB.

- `formatEnable` – If set to NPF_FALSE no formatting is performed as a result of calling this function, the parameters 'header', 'numEntries' and 'formatArray' are not valid and the LFB is returning operation to default mode (which is: take the txAddress, from the ATM Configuration Manager, as a header with no formatting applied). This is the way to remove all formatting that might have been applied in earlier function calls. If set to NPF_TRUE formatting is performed as specified by the parameters 'header', 'numEntries' and 'formatArray'.

- `header` – If set to NPF_FALSE the txAddress (from the ATM Configuration Manager) is placed as a trailer in relation to the packet SDU. If set to NPF_TRUE the txAddress (from the ATM Configuration Manager) is placed as a header in relation to the packet SDU. See **Figure 3** for a better understanding.

- `numEntries` – Number of metadata/strings to format into the header/trailer part of the Backplane PDU.

- `formatArray` – Pointer to an array of metadata/string structures that shall be formatted. The order of the formatting is specified with the 'formatOrder' parameter inside the NPF_F_SwAddrGenATM_Format_t structure.

## *Output Parameters*

None

## *Return Values*

- `NPF_NO_ERROR` - The operation is in progress.
- `NPF_E_UNKNOWN` - The format operation did not complete successfully due to problems encountered when handling the input parameters.
- `NPF_E_BAD_CALLBACK_HANDLE` - The format operation was not completed because the callback handle was invalid.
- `NPF_E_FUNCTION_NOT_SUPPORTED` - The function call is not supported by this implementation.

## *Asynchronous Response*

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

- `NPF_NO_ERROR` – Operation completed successfully.
- `NPF_E_SWADDRGENATM_INVALID_BLOCK_ID` – LFB ID is not an ID of LFB that has Switch Address Generator ATM functionality.

The error code is returned in the error field.

# 6    References

[SWAPICON]    [Software API Conventions Revision 2](#)
[ATMLFBARC]  [ATM Software API Architecture Framework Revision 1.0](#)
[ATMMGR]        [ATM Configuration Manager Functional API Revision 1.0](#)

## APPENDIX A

### A.1.  <u>API CALL CAPABILITIES</u>

This table lists the mandatory, conditional and optional functions to be present in an implementation of this FAPI.

The registration/deregistration functions are required if any of the optional APIs is implemented by the LFB.

Note: M = mandatory; O = optional; C = Conditional

Table 8 API Call Capabilities

| API function | Function Required |
|---|---|
| NPF_F_SwAddrGenATM_Register | C |
| NPF_F_SwAddrGenATM_Deregister | C |
| NPF_F_SwAddrGenATM_LFB_AttributesQuery | O |
| NPF_F_SwAddrGenATM_Format | O |

### A.2.  <u>HEADER FILE – NPF_F_SWADDRGENATM.H</u>

```
/*
 * This header file defines typedefs, constants and structures
 * for the NP Forum Switch Address Generator ATM Functional API
 */
#ifndef __NPF_F_SW_ADDRESS_GENERATOR_ATM_H__
#define __NPF_F_SW_ADDRESS_GENERATOR_ATM_H__

#ifdef __cplusplus
extern "C" {
#endif

#define NPF_F_SWADDRGENATM_LFB_TYPE    45

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

#define NPF_SWADDRGENATM_BASE_ERR (NPF_F_ATMHDRGEN_LFB_TYPE * 100)
#define NPF_E_SWADDRGENATM_INVALID_BLOCK_ID\
                            (NPF_SWADDRGENATM_BASE_ERR + 0)


/***************************************************************
 * Enumerations and types for Switch Address Generator ATM LFB  *
 ***************************************************************/
/*  This type defines the input to apply the formatting on. */
typedef enum {
   NPF_F_SWADDRGENATM_ATM_SDU_IN = 0,             /* ATM SDU input. */
   NPF_F_SWADDRGENATM_AAL5_CPCS_IF_DATA_IN = 1, /* AAL5 packet input. */
   NPF_F_SWADDRGENATM_AAL2_CPS_SDU_IN = 2,      /* AAL2 CPS SDU input. */
   NPF_F_SWADDRGENATM_AAL2_SSSAR_SDU_IN = 3     /* AAL2 SSSAR SDU input. */
} NPF_F_SwAddrGenATM_Input_t;
```

```
/* This type defines the metadata on the ATM_SDU_IN input that can be
 * formatted and put into the output header/trailer part of the
 * Backplane PDU */
typedef enum {
   NPF_F_SWADDRGENATM_ATM_SDU_PTI = 0,    /* Payload Type Identifier */
   NPF_F_SWADDRGENATM_ATM_SDU_LP = 1,     /* Loss Priority */
   NPF_F_SWADDRGENATM_ATM_SDU_VCI = 2     /* Virtual Channel Identifier */
} NPF_F_SwAddrGenATM_AtmSduMetadata_t;


/* This type defines the metadata on the AAL5_CPCS_IF_DATA_IN input
 * that can be formatted and put into the output header/trailer
 * part of the Backplane PDU */
typedef enum {
   NPF_F_SWADDRGENATM_AAL5_CPCS_LP = 0,    /* CPCS Loss Priority */
   NPF_F_SWADDRGENATM_AAL5_UUI = 1,        /* CPCS UUI */
   NPF_F_SWADDRGENATM_AAL5_FRAMELENGTH = 2, /* Length of the packet */
   NPF_F_SWADDRGENATM_AAL5_BOUND_IF = 3,   /* Handle of the child interface */
   NPF_F_SWADDRGENATM_AAL5_RCV_STATUS = 4 /* Type of error in reassembly */
} NPF_F_SwAddrGenATM_Aal5CpcsMetadata_t;


/* This type defines the metadata on the AAL2_CPS_SDU_IN input
 * that can be formatted and put into the output header/trailer
 * part of the Backplane PDU */
typedef enum {
   NPF_F_SWADDRGENATM_AAL2_CPS_UUI = 0,    /* CPS UUI */
   NPF_F_SWADDRGENATM_AAL2_CPS_LI = 1,     /* CPS LI */
   NPF_F_SWADDRGENATM_AAL2_BOUND_IF = 2    /* Handle of higher layer i/f */
} NPF_F_SwAddrGenATM_Aal2CpsSduMetadata_t;


/* This type defines the metadata on the AAL2_SSSAR_SDU_IN input
 * that can be formatted and put into the output header/trailer
 * part of the Backplane PDU */
typedef enum {
   NPF_F_SWADDRGENATM_AAL2_SSSAR_UUI = 0, /* UUI of SAR SDU */
   NPF_F_SWADDRGENATM_AAL2_SSSAR_LI = 1   /* LI of SAR SDU */
} NPF_F_SwAddrGenATM_Aal2SssarSduMetadata_t;

/* Defines the string that can be formatted and put into the
 * output header/trailer part of the Backplane PDU. */
typedef struct {
   NPF_uint32_t   stringLength;      /* Length of the String. */
   /* Pointer to an array of string octets. */
   NPF_uint8_t    *string;
} NPF_F_SwAddrGenATM_String_t;

/* This type defines the metadata format structure. */
 typedef struct {
   /* Specifies which of the four inputs to apply the formatting to. */
   NPF_F_SwAddrGenATM_Input_t         inputToFormat;
   /* Specifies if the formatting operation shall use an input metadata
    * or a static string. If set to NPF_TRUE the 'metadataIn' is used,
    * if set to NPF_FALSE 'string' is used. */
   NPF_boolean_t                      metadata;
   /* Specifies if the metadata/string shall be overwriting existing bits
    * of the txAddress or be inserted at a specified bit position. */
   NPF_boolean_t                      overwrite;
   union {
```

```
        /* The metadata input to be used for the formatting. */
        union {
            NPF_F_SwAddrGenATM_AtmSduMetadata_t atmSduMetaIn;
            NPF_F_SwAddrGenATM_Aal5CpcsMetadata_t aal5CpcsMetaIn;
            NPF_F_SwAddrGenATM_Aal2CpsSduMetadata_t aal2CpsSduMetaIn;
            NPF_F_SwAddrGenATM_Aal2SssarSduMetadata_t aal2SssarSduMetaIn;
        }u1;
        /* The string to be used for the formatting. */
        NPF_F_SwAddrGenATM_String_t   string;
    }u2;
    /* The start bit of where the input metadata/string shall be overwriting
     * existing bits or be inserted. See Figure 5 and Figure 6 for an
     * overview of the bit numbering definition. */
    NPF_uint32_t                    startBit;
    /* The bit length of the input metadata/string that shall be overwriting
     * existing bits or be inserted. See Figure 5 and Figure 6 for an
     * overview of the bit numbering definition. For an insert operation
     * the bitlength MUST be a multiple of 8 (i.e. 8, 16, ..). An overwrite
     * operation has no such limitations. */
    NPF_uint32_t                    bitLength;
    /* This parameter specifies the order of formatting. If numEntries = 1,
     * the formatOrder MUST be set to '1'. If numEntries > 1, the formatOrder
     * must be set to '1, 2, 3, ... , numEntries' respectively. The format
     * operation with formatOrder = 1 SHALL always be executed first, the
     * format operation with formatOrder = 2 SHALL always be executed second
     * and so on. */
    NPF_uint32_t                    formatOrder;
} NPF_F_SwAddrGenATM_Format_t;

/* Attributes Query Response. */
typedef struct {
    NPF_uint32_t   maxBpls;        /* Maximum possible BP links   */
    NPF_uint32_t   curNumBpls;     /* Current number of BP links  */
    NPF_uint32_t   maxTxAddrLength; /* Maximum possible TX Address length */
} NPF_F_SwAddrGenATM_LFB_AttrQueryResponse_t;

/*
 * Completion Callback Types, to be found in the callback
 * data structure, NPF_F_SwAddrGenATM_CallbackData_t.
 */
typedef enum NPF_F_SwAddrGenATM_CallbackType {
    NPF_F_SWADDRGENATM_ATTR_QUERY = 1,
} NPF_F_SwAddrGenATM_CallbackType_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_SwAddrGenATM_ErrorType_t error;/* Error code for response   */
    union {
        /* NPF_F_SwAddrGenATM_LFB_AttributesQuery() */
        NPF_F_SwAddrGenATM_LFB_AttrQueryResponse_t  lfbAttrQueryResponse;
    } u;
} NPF_F_SwAddrGenATM_AsyncResp_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_ATM_AsyncResponse_t structure, along with any other information
 */
typedef struct {
   NPF_F_SwAddrGenATM_CallbackType_t type; /* Which function called?   */
   NPF_IN NPF_BlockId_t          blockId;/* ID of LFB generating callback */
   NPF_F_SwAddrGenATM_AsyncResp_t  resp;/* response structure */
} NPF_F_SwAddrGenATM_CallbackData_t;

/****************************************************************************
 * Switch Address Generator ATM LFB Registration/De-registration Functions *
 ****************************************************************************/
typedef void (*NPF_F_SwAddrGenATM_CallbackFunc_t) (
   NPF_IN NPF_userContext_t                   userContext,
   NPF_IN NPF_correlator_t                    correlator,
   NPF_IN NPF_F_SwAddrGenATM_CallbackData_t   data);

NPF_error_t NPF_F_SwAddrGenATM_Register(
   NPF_IN NPF_userContext_t                   userContext,
   NPF_IN NPF_F_SwAddrGenATM_CallbackFunc_t   callbackFunc,
   NPF_OUT NPF_callbackHandle_t               *callbackHandle);

NPF_error_t NPF_F_SwAddrGenATM_Deregister(
   NPF_IN NPF_callbackHandle_t    callbackHandle);

/************************************************************
 * Switch Address Generator ATM LFB optional functions       *
 ************************************************************/
NPF_error_t NPF_F_SwAddrGenATM_LFB_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);

NPF_error_t NPF_F_SwAddrGenATM_Format(
   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,
   NPF_IN NPF_boolean_t                      formatEnable,
   NPF_IN NPF_boolean_t                      header,
   NPF_IN NPF_uint32_t                       numEntries,
   NPF_IN NPF_F_SwAddrGenATM_Format_t        *formatArray);

#ifdef __cplusplus
}
#endif
#endif /* __NPF_F_SW_ADDRESS_GENERATOR_ATM_H__ */
```

## APPENDIX B    <u>ACKNOWLEDGEMENTS</u>

**Working Group Chair**:

**Alex Conta, Transwitch,** aconta@txc.com

**Task Group Chair**:

**Per Wollbrand, Ericsson,** per.wollbrand@ericsson.com

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

The list is in alphabetical order of last names:

**Pat Carr, Wintegra**

**Pål Dammvik, Ericsson**

**Patrik Herneld, Ericsson**

**Alistair Munro, U4EA**

**John Renwick, Agere Systems**

**Vedvyas Shanbhogue, Intel**

**Roger Smith, Wintegra**

**Per Wollbrand, Ericsson**

## APPENDIX C    LIST OF COMPANIES BELONGING TO NPF DURING APPROVAL PROCESS

Table 9 List of Companies

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