M3UA API & Users' Manual

Version 3.0.6

1 Introduction

The basis of M3UA protocol implementation is RFC 3332 available at IETF website (http://www.ietf.org). The entire mandatory & almost all the optional functionalities are available with this software.

This manual explains various API Primitives (Application Programming Interface Primitives) provided by M3UA protocol implementation (referred as just M3UA in the remaining document). It describes API parameters as well as the sequence in which API's should be invoked.

This manual should be used in conjunction with the M3UA sample application to understand the overall functioning of the M3UA library. The aim of this manual is to explain all API services of the M3UA library. The sample application puts them into use and thus serves as a live working example for M3UA library.

2 API Operations

API's that are used for configuration purpose, perform operations on M3UA components like ASP, SGP, AS etc. Operations are referred as M3_ADD, M3_DELETE, M3_GET and M3_MODIFY in the rest of the document. Following is the description of these operations.

- **M3_ADD**: - Add a M3UA object (AS, ASP, SGP, SG etc.) to the M3UA.
- **M3_DELETE**: - Delete an already provisioned M3UA object from M3UA.
- **M3_GET**: - Request configuration of an already provisioned M3UA object.
- **M3_MODIFY**: - Change the configuration of an already provisioned M3UA object.

Depending on the operations supported by a configuration API, its parameters are divided into different structures in a union. Based on the type of operation requested, only required parameters need to be filled. *For example, while adding an ASP to the M3UA, only "add" structure of union m3ua_asp_t must be filled.*

3 API Return Values

Every API returns "–1" in case of failure and a non-negative value after successful execution.

4 M3UA Library Installation

You must have downloaded the source code for M3UA in the form of gzipped tar. Name of this gzipped tar is m3ua_vX_Y_Z.tar.gz. Just unzip this file using command `` `gzip -d m3ua_vX_Y_Z.tar.gz` ``'.

This will produce a file with name m3ua_vX_Y_Z.tar. Untar this file using command `` `tar xvf m3ua_vX_Y_Z.tar' ``.

Now you have complete source code for M3UA installed in the directory `./m3ua'. Change to directory `./m3ua' using command '**`cd ./m3ua`**' to browse through the code.

4.1 Building the M3UA library

To build M3UA library from the source code follow the steps below:

1. Change to directory containing the source code for M3UA. Now change to directory `./m3ua/src'.

2. [Optional] Skip this step and jump to step 3 if you are not interested in building dependencies. Type `make depend' to build the dependencies.

3. Type `make' to build the library libm3ua.a. This library is present in ./m3ua/src' directory.

4. You can remove the program binaries and object files from the source code directory by typing `make clean'.

4.2 Building the sample application

To build M3UA sample applications from the source code follow the steps below:

1. Save file m3ua_comm_app_vX_Y_Z.tar.gz in the same directory where m3ua library (./m3ua) has been extracted. Or, in the parent directory of "./m3ua" directory.

2. Untar and unzip the sample application using following commands:
    - `untar –xvf ./m3ua_comm_app_vX_Y_Z.tar.gz`
    - `gzip –d ./m3ua_comm_app_vX_Y_Z.tar`

3. Change to directory "./comm" and type following command to build sample applications:
    - If SCTP (LKSCTP 1.0.6 is already installed on the system), type following command:
        - `make sctp`

- If SCTP is not installed and you would like to build sample applications using UDP, then give following command
    - `make asp1 asp2 sgp1`

4. You can remove the program binaries and object files from the source code directory by typing `` `make clean'``.

5. Sample applications have been designed to demonstrate both ASP-SGP model and IPSP-IPSP model. Sample applications "asp1" and "sgp1" operate in ASP-SGP mode. At the same time "asp1" and "asp2" operate in IPSP-IPSP mode.

4.3 Compilers and Options

1) We have used only `` `gcc'`` compiler presently due to its wide spread popularity.
2) Use option `__LITTLE_ENDIAN__` while compiling for little endian architectures like Intel Pentium machines. For big endian architectures no option is required. M3UA is expected to perform better on big endian architectures.

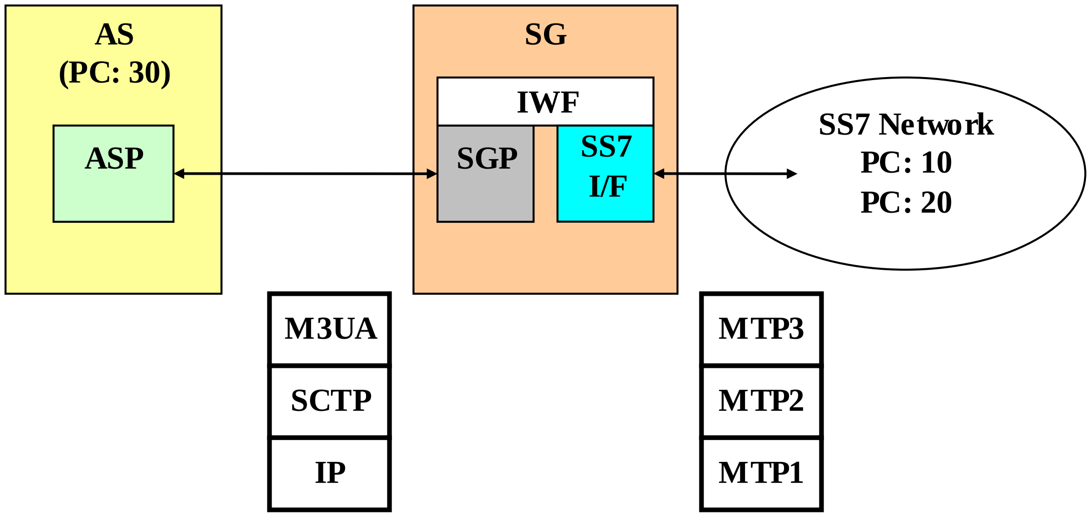
5 Example Scenarios & Sequence of API calls

This section lists example scenarios where M3UA protocol may be used. The M3UA protocol supports almost all the possible scenarios and this section lists a sub-set of those scenarios.

5.1 Conventional Scenario – AS to SG Communication

Following figure shows the case where M3UA would be used for transporting SS7 signaling between Application Server in IP network and signaling points in SS7 network. In present example, signaling points with PC 10 and 20 in SS7 network would communicate with application server in IP network with PC 30.

Before M3UA configuration is started at any node, it is assumed that SCTP has been configured and SCTP associations are already prepared.



Note:-

AS is Application Server
ASP is Application Server Process
IWF is Inter Working Function
SS7 I/F is SS7 Interface
SG is Signaling Gateway
SGP is Signaling Gateway Process

5.1.1 Configuring M3UA at Signaling Gateway

Following is the sequence of APIs that need to be invoked to configure M3UA at signaling gateway.

Note:- LM is Layer Management

1. **M3UA_INIT (LM→M3UA)** – Initialize the M3UA protocol layer
2. **M3UA_NWAPP (LM→M3UA)** – Configure the network appearance (network ID / network standard)
3. **M3UA_R_ASP (LM→M3UA)** – Configure remote ASP
4. **M3UA_R_AS (LM→M3UA)** – Configure remote AS
5. **M3UA_SGP (LM→M3UA)** – Configure local SGP
6. **M3UA_USER (LM→M3UA)** – Create local user (in this case IWF would register itself as the user with M3UA).
7. **M3UA_CONN (LM→M3UA)** – Create a logical M3UA connection between local SGP and remote ASP. There is a one to one mapping between a SCTP association and a M3UA connection.
8. **M3UA_CONN_STATE (LM→M3UA)** – Change state of M3UA connection between SGP and ASP to established. This is done to indicate to M3UA layer that SCTP association is ready to be used.

9. **M3UA_MGMT_NTFY (M3UA → LM)** – Once M3UA indicates to layer management that remote ASP is ACTIVE to process SS7 messages, data transfer between SS7 network and application server in IP network may begin. Now upper layers of SS7 stack may be provisioned.
10. **M3UA_USER_NTFY (M3UA→USER)** – Audit indication is given from M3UA layer to the IWF. IWF checks the availability of signaling points in the SS7 network and invokes M3UA_PAUSE or M3UA_RESUME or M3UA_STATUS API. This may happen anytime during remote ASP is in ACTIVE state.
11. **M3UA_TRANSFER (USER→M3UA)** – For transferring SS7 messages from SS7 network to application server in IP network.
12. **M3UA_USER_NTFY (M3UA→USER)** – M3UA gives SS7 messages received from application server in IP network to the IWF through this primitive.

5.1.2 Configuring M3UA at Application Server

Following is the sequence of APIs that need to be invoked to configure M3UA at application server.

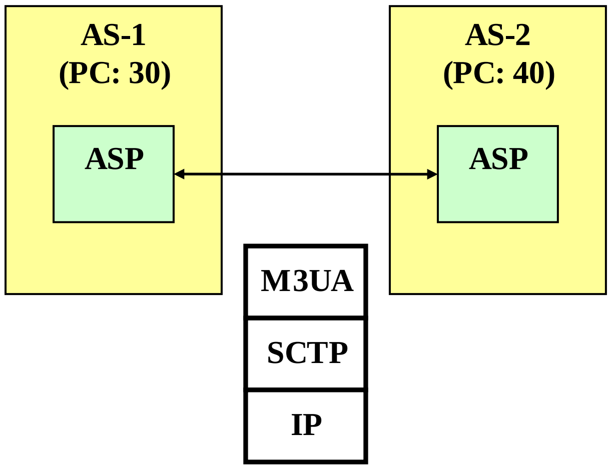
1. **M3UA_INIT (LM→M3UA)** – Initialize the M3UA protocol layer
2. **M3UA_NWAPP (LM→M3UA)** -- Configure the network appearance (network ID / network standard)
3. **M3UA_R_SGP (LM→M3UA)** – Configure remote SGP
4. **M3UA_SG (LM→M3UA)** – Configure remote Signaling Gateway (SG) server
5. **M3UA_AS (LM**→M3UA) – Configure local Application server
6. **M3UA_ASP (LM→M3UA)** – Configure local Application Server Process
7. **M3UA_USER (LM→M3UA)** – Configure a particular M3UA user (M3UA users like SCCP, ISUP etc)
8. **M3UA_ROUTE (LM→M3UA)** – Add route towards PC 10
9. **M3UA_ROUTE (LM→M3UA)** – Add route towards PC 20
10. **M3UA_CONN (LM→M3UA)** – Add M3UA logical connection between the local ASP and the remote SGP. There is a one to one mapping between a SCTP association and a M3UA connection.
11. **M3UA_CONN_STATE (LM→M3UA)** – Change state of M3UA connection between SGP and ASP to established. This is done to indicate to M3UA layer that SCTP association is ready to be used.
12. **M3UA_ASP_STATE (LM→M3UA)** – Change local ASP state to INACTIVE
13. **M3UA_MGMT_NTFY (M3UA → LM)** –M3UA indicates to layer management that local ASP is INACTIVE
14. **M3UA_ASP_STATE (LM→M3UA)** – Change local ASP state to ACTIVE
15. **M3UA_MGMT_NTFY (M3UA → LM)** –M3UA indicates to layer management that local ASP is ACTIVE and ready to process SS7 messages. Now upper layers of SS7 stack may be provisioned.
16. **M3UA_AUDIT (LM→M3UA)** – Initiate auditing of point codes 10 & 20 to check their availability. Layer management initiates auditing on behalf of a particular M3UA user.

17. **M3UA_USER_NTFY (M3UA→USER)** – M3UA gives RESUME or PAUSE or STATUS indication to the users registered with it.
18. **M3UA_TRANSFER (USER→M3UA)** – For transferring SS7 messages from SS7 network to application server in IP network.
19. **M3UA_USER_NTFY (M3UA→USER)** – M3UA gives SS7 messages received from application server in IP network to the IWF through this primitive.

5.2 IPSP Scenario – AS to AS Communication

Following figure shows the case where M3UA would be used for transporting SS7 signaling between two Application Servers in the IP network. Following figure illustrates this example.

Before M3UA configuration is started at any node, it is assumed that SCTP has been configured and SCTP associations are already prepared.



Note:-

AS is Application Server
ASP is Application Server Process
IWF is Inter Working Function
SS7 I/F is SS7 Interface
SG is Signaling Gateway
SGP is Signaling Gateway Process

5.2.1 Configuring M3UA at Application Server – AS-1

Following is the sequence of APIs that need to be invoked to configure M3UA at application server (AS-1).

1. **M3UA_INIT (LM→M3UA)** – Initialize the M3UA protocol layer

2. **M3UA_NWAPP (LM→M3UA)** – Configure the network appearance (network ID / network standard)
3. **M3UA_R_ASP (LM→M3UA)** – Configure remote ASP
4. **M3UA_R_AS (LM→M3UA)** – Configure remote AS
5. **M3UA_AS (LM→M3UA)** – Configure local AS
6. **M3UA_ASP (LM→M3UA)** – Configure local ASP
7. **M3UA_USER (LM→M3UA)** – Configure a particular M3UA user (M3UA users like SCCP, ISUP etc)
8. **M3UA_CONN (LM→M3UA)** – Create a logical M3UA connection between local SGP and remote ASP. There is a one to one mapping between a SCTP association and a M3UA connection.
9. **M3UA_CONN_STATE (LM→M3UA)** – Change state of M3UA connection between SGP and ASP to established. This is done to indicate to M3UA layer that SCTP association is ready to be used.
10. **M3UA_MGMT_NTFY (M3UA→LM)** – Once M3UA indicates to layer management that remote ASP is ACTIVE to receive SS7 signaling, data transfer between SS7 network and application server in IP network may begin. Now upper layers of SS7 stack may be provisioned.
11. **M3UA_TRANSFER (USER→M3UA)** – For transferring SS7 messages from SS7 network to application server in IP network.
12. **M3UA_USER_NTFY (M3UA→USER)** – M3UA gives SS7 messages received from application server in IP network to the IWF through this primitive.

5.2.2 Configuring M3UA at Application Server – AS-2

Following is the sequence of APIs that need to be invoked to configure M3UA at application server (AS-2).

1. **M3UA_INIT (LM→M3UA)** – Initialize the M3UA protocol layer
2. **M3UA_NWAPP (LM→M3UA)** – Configure the network appearance (network ID / network standard)
3. **M3UA_R_ASP (LM→M3UA)** – Configure remote ASP
4. **M3UA_R_AS (LM→M3UA)** – Configure remote AS
5. **M3UA_AS (LM→M3UA)** – Configure local AS
6. **M3UA_ASP (LM→M3UA)** – Configure local ASP
7. **M3UA_USER (LM→M3UA)** – Configure a particular M3UA user (M3UA users like SCCP, ISUP etc)
8. **M3UA_CONN (LM→M3UA)** – Create a logical M3UA connection between local SGP and remote ASP. There is a one to one mapping between a SCTP association and a M3UA connection.
9. **M3UA_CONN_STATE (LM→M3UA)** – Change state of M3UA connection between SGP and ASP to established. This is done to indicate to M3UA layer that SCTP association is ready to be used.
10. **M3UA_ASP_STATE (LM→M3UA)** – Change local ASP state to INACTIVE

11. **M3UA_MGMT_NTFY (M3UA → LM)** –M3UA indicates to layer management that local ASP is INACTIVE
12. **M3UA_ASP_STATE (LM→M3UA)** – Change local ASP state to ACTIVE
13. **M3UA_MGMT_NTFY (M3UA → LM)** –M3UA indicates to layer management that local ASP is ACTIVE and ready to process SS7 messages. Now upper layers of SS7 stack may be provisioned.
14. **M3UA_TRANSFER (USER→M3UA)** – For transferring SS7 messages from SS7 network to application server in IP network.
15. **M3UA_USER_NTFY (M3UA→USER)** – M3UA gives SS7 messages received from application server in IP network to the IWF through this primitive.

6 Timing Management for M3UA Library

M3UA library manages timers internally and also takes appropriate actions whenever any timer expires. But timer ticks need to be supplied to the M3UA library from an external source. A timer tick may be supplied to M3UA library using M3UA_TIMER_CHECK API. This API is discussed later in the document.

As M3UA timers have a granularity in seconds, so it is recommended that at least one timer tick is provided to the M3UA library every second. More than one timer tick may also be provided based on the application design.

Supplying a timer tick to M3UA library is similar to passing an external event like SCTP message or SS7 user message. All the external events must be passed to the M3UA library using a single thread. In simpler words, more than one thread must not be used to pass external events to M3UA library. This may lead to M3UA state machine going out of synchronization. For example: one thread passes an ASPAC_ACK message and other thread passes ASPM timer expiry event, then M3UA state machine may go out of sync because these two events can never happen simultaneously.

7 Collecting Logs from M3UA Library

M3UA library logs messages using following two C macros:

1. M3TRACE – For logging messages
2. M3HEX – For logging hexadecimal contents

An application designer may port above two macros as per convenience to collect traces (logs) at desired location. These two macros are defined in include file “m3uaTraceMgr.h”. Presently, these macros are using standard C IO function (printf) for logging messages directly to the screen.

8 API between M3UA and SCTP

This section describes API interface between M3UA and SCTP layers. Complete details of these APIs may be found in document later.

8.1 Managing SCTP Association States

It is assumed that before M3UA is configured, SCTP associations are already provisioned and ready to use. Any change in SCTP association state needs to be updated in the M3UA layer through M3UA_CONN_STATE API. Following are the SCTP association states that need to be updated in the M3UA layer:

- ASSOCIATION SHUTDOWN
- ASSOCIATION ESTABLISHED

- ASSOCIATION CONGESTED

8.2 Sending Data to SCTP

M3UA layer would send any messages to SCTP layer using the API M3UA_SENDMSG. It is responsibility of M3UA application developer to write this API using services provided by the underlying transport layer [SCTP in this case].

8.3 Receiving Data from SCTP

Whenever any data is received on a particular SCTP association, it needs to be passed to the M3UA layer. This can be done through M3UA_RECVMSG API. This API needs to be invoked with association ID on which data is received and message details [message content and message size].

9 API [Application Programming Interface] Usage

Following is the list (in given sequence) of include files that shall be added in the application source files which would make use of M3UA API's. **We strongly recommend you to have a look in the sample application source code also that is present in directory './comm'. This would give you in-depth idea of using M3UA API's.**

```
#include <m3ua_defines.h>
#include <m3ua_types.h>
#include <m3ua_api.h>
#include <m3ua_errno.h>
```

Above *include files* contain structures, enumerations, compile time defined values, error numbers and various other data structures that would help in writing application based on M3UA library.

This section of the document describes API's in detail, and services provided by each of them. Please see section "***Data Structures Associated with API Primitives"*** for source code of structures used along with API primitives.

10 Scaling M3UA Library

Different applications would have different requirements from M3UA library in terms of number of managed objects it supports. Also, number of memory buffers may be adjusted as per traffic requirements. This scaling process may be done before M3UA is compiled.

Define Name	Include File	Default Value	Description
M3_MAX_AS	m3ua_defines.h	8	Maximum no. of Application servers
M3_MAX_ASP	m3ua_defines.h	64	Maximum no. of Application Server Processes
M3_MAX_R_ASP	m3ua_defines.h	128	Maximum no. of Remote Application Server Processes
M3_MAX_R_SGP	m3ua_defines.h	32	Maximum no. of Remote Signaling Gateway Processes
M3_MAX_SGP	m3ua_defines.h	4	Maximum no. of Signaling Gateway Processes
M3_MAX_R_AS	m3ua_defines.h	64	Maximum no. of remote Application Servers
M3_MAX_SG	m3ua_defines.h	8	Maximum no. of signaling gateways
M3_MAX_CONN	m3ua_defines.h	128	Maximum no. of M3UA connections (or SCTP associations)
M3_MAX_ASSOCID	m3ua_defines.h	512	Maximum value of SCTP association ID
M3_MAX_NWAPP	m3ua_defines.h	8	Maximum no. of network appearances
M3_MAX_USR	m3ua_defines.h	32	Maximum no. of M3UA users
M3_MAX_ROUTES_PER_SG	m3ua_defines.h	64	Maximum no. of routes (point codes) reachable through signaling gateway
M3_MAX_TIMERS	m3ua_defines.h	256	Total no. of timers supported by M3UA library
M3_NUM_32BYTE_BUFS	m3ua_defines.h	64	Total no. of 32 byte memory buffers
M3_NUM_64BYTE_BUFS	m3ua_defines.h	256	Total no. of 64 byte memory buffers
M3_NUM_128BYTE_BUFS	m3ua_defines.h	256	Total no. of 128 byte memory buffers
M3_NUM_256BYTE_BUFS	m3ua_defines.h	256	Total no. of 256 byte memory buffers

M3_NUM_512BYTE_BUFS	m3ua_defines.h	128	Total no. of 512 byte memory buffers
M3_NUM_1024BYTE_BUFS	m3ua_defines.h	64	Total no. of 1024 byte memory buffers
M3_NUM_2048BYTE_BUFS	m3ua_defines.h	32	Total no. of 2048 byte memory buffers
M3_NUM_4096BYTE_BUFS	m3ua_defines.h	16	Total no. of 4096 byte memory buffers
M3_NUM_8192BYTE_BUFS	m3ua_defines.h	16	Total no. of 8192 byte memory buffers

For example; if an application requires connecting with only two remote SGP, then the value of M3_MAX_R_SGP may be reduced to 2. Similarly any other value above may be fine tuned as per the requirements.

11 Layer Management API (LM→M3UA)

Following are Layer Management (LM) APIs that are used to configure M3UA layer.

11.1 M3UA_INIT

Prototype	m3_s32 m3ua_init(void);	
Parameters		
Operations		
Description	This API is used by M3UA to initialize internal data structures. This API must be invoked before invoking any other API. An application developer may scale internal data structures as per the requirement. Size of internal structures are defined in the include file “m3ua_defines.h”.	
Restrictions/Bugs	None	
Pre-requisites		
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_init	

11.2 M3UA_NWAPP

Prototype	m3_s32 m3ua_nwapp(m3_u8 oprn, m3ua_nwapp_t *pInf);	
Parameters	oprn	Operation to be performed using this API. Presently only M3_ADD operation is supported.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_nwapp_conf_t    info;`
		`    } add;`

		`    struct {`
		`        m3_nwapp_conf_t    info;`
		`    } del;`
		`} m3ua_nwapp_t;`
		
		`typedef struct {`
		`    m3_u32                 nw_app;`
		`    m3_standard_t          standard;`
		`} m3_nwapp_conf_t;`
		
		`typedef enum {`
		`    M3_STD_ANSI,`
		`    M3_STD_ITU`
		`} m3_standard_t;`

	nw_app	Value of network appearance. It is an unsigned integer value and can take any value between 0x0 to 0xFF FFFFFE.
	standard	SS7 Standard (SS7 variant - ITU/ANSI) associated with the network appearance. It is an enumerated value and takes following values: M3_STD_ANSI (0) for ANSI standard M3_STD_ITU (1) for ITU standard
Operations	M3_ADD	
Description	This API is used to ADD (or provision) network appearance associated with the SS7 networks to be served by M3UA. Traffic related to the SS7 network is identified using this network appearance. It is important that at least once NWAPP is configured so that M3UA library can use the associated SS7 standard (or variant, example: ITU or ANSI) for encoding and decoding DATA messages.	
Restrictions/Bugs	This API cannot be used to delete an already configured network appearance.	
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_nwapp	

11.3 M3UA_ASP

Prototype	m3_s32 m3ua_asp(m3_u32 aspId, m3_u8 oprn, m3ua_asp_t *pInf);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_asp_conf_t info;`
		`    } add;`
		`    struct {`
		`        m3_asp_conf_t info;`
		`    } get;`
		`    struct {`
		`        m3_asp_confname_t    confname;`
		`        m3_asp_conf_t        info;`

		```
		        m3_u32                  as_id;
		        m3_u32                  ras_id;
		        m3_u32                  rasp_id;
		    } modify;
		} m3ua_asp_t;

		typedef struct
		{
		    m3_u32                  m3asp_id;
		    m3_u32                  sctp_ep_id;
		    m3_bool_t
		as_list[M3_MAX_AS];
		    m3_u32                  def_nwapp;
		    struct {
		        m3_bool_t
		asp_list[M3_MAX_R_ASP];
		    } r_as_inf[M3_MAX_R_AS];
		} m3_asp_conf_t;
		```
	aspId	Identifier of ASP on which operation is to be performed. It is an unsigned integer value. 16 Most Significant Bits of this parameter should always be set to 0. The least significant 16 bits can be assigned any value between between 0 and (M3_MAX_ASP – 1). The value for M3_MAX_ASP is defined in “m3ua_defines.h” include file.
	m3asp_id	ASP identifier to be used in the messages (ASPUP and NTFY) exchanged between M3UA peer entities. It is an unsigned integer value and can take any value between 0x0 to 0xFFFFFFFE. This value may be excluded from M3UA messages by using M3UA_CONN_OPT API.
	sctp_ep_id	SCTP Endpoint identifier for the ASP. It is an unsigned integer value. As every ASP is associated with an SCTP endpoint (combination of IP Address/SCTP Port). This value is just for convenience of application designer to associate an SCTP end point with an ASP.
	as_list[]	List of Application Servers served by this ASP. For example, if application server “asId” is served by this ASP, then as_list[asId] should be set to M3_TRUE else it should be set to M3_FALSE.
	def_nwapp	Default network appearance to be assumed while processing a message received without a network appearance at the ASP. This is an unsigned integer value. The network appearance specified in this parameter must be configured through M3UA_NWAPP API.
	r_as_inf[]	Remote Application Server related information specific to this ASP. This parameter specifies which remote ASP’s are serving this particular remote AS. As it is possible to configure several ASP’s in the M3UA library, so every ASP has a separate

		configuration of remote application servers.
	asp_list[]	List of remote ASP serving the remote AS. This is like making an authorized list of remote ASP's that are allowed to process traffic for specified remote AS. For example, if remote application server process "rasp" is serving remote application server "ras" then r_as_inf[ras].asp_list[rasp] should be set to M3_TRUE else it should be set to M3_FALSE.
	confname	Name of ASP configuration to be modified. Following are possible modifications for ASP configuration: • M3_ASP_M3ASP_ID – Modify M3UA ASP Identifier. • M3_ASP_NWAPP – Modify default NWAPP associated with this ASP • M3_ASP_ADD_AS – Add an application server to ASP configuration so that this application server is served by this ASP. • M3_ASP_DEL_AS – Remove an application server from ASP, so that ASP is no more serving the application server. • M3_ASP_ADD_R_ASP – Specify a remote ASP that will process traffic for a specified remote AS. • M3_ASP_DEL_R_ASP – Specified remote ASP will no more process traffic for specified remote AS.
	as_id	Add/delete an application server from the ASP configuration when configuration name parameter is set to M3_ASP_ADD_AS/M3_ASP_DEL_AS.
	ras_id	Used to modify ASP configuration when configuration name parameter is set to M3_ASP_ADD_R_ASP or M3_ASP_DEL_R_ASP. A remote AS should be specified while adding/deleting a remote ASP to the ASP configuration.
	rasp_id	Used to modify ASP configuration when configuration name parameter is set to M3_ASP_ADD_R_ASP or M3_ASP_DEL_R_ASP. The remote ASP specified to be added or deleted is specified by this parameter. This remote ASP is serving remote AS specified by the parameter "ras_id".
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get/modify ASP configuration in the M3UA. This ASP receives traffic for the AS it serves. The ASP also routes traffic originating from these AS to configured remote AS & SG. The ASP should be brought into ACTIVE state before it starts routing any traffic.	
Restrictions/Bugs	All the remote ASP and remote AS that are part of the ASP configuration should be added before adding the ASP. All the connections from this ASP must be deleted before deleting this ASP.	
Pre-requisites	M3UA_INIT, M3UA_NWAPP, M3UA_R_ASP, M3UA_R_AS, M3UA_AS	
Code Sample	Please refer source file comm/src/confasp1.c, function name	

	um3_m3ua_asp

11.4 M3UA_AS

Prototype	M3_s32 m3ua_as(m3_u32 as_id, m3_u8 oprn, m3ua_as_t *p_inf);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_as_conf_t info;`
		`    } add;`
		`    struct {`
		`        m3_as_conf_t info;`
		`    } get;`
		`    struct {`
		`        m3_as_confname_t    confname;`
		`        m3_as_conf_t        info;`
		`    } modify;`
		`} m3ua_as_t;`
		
		`typedef struct`
		`{`
		`    m3_u32                rtctx;`
		`    m3_rk_inf_t           rkey;`
		`} m3_as_conf_t;`
		
		`typedef struct`
		`{`
		`    m3_traffic_mode_t     trfmode;`
		`    m3_u32                nw_app;`
		`    m3_u8                 num_rtparam;`
		`    m3_rk_elements_t`
		`rtparam[M3_MAX_DPC_PER_RK];`
		`} m3_rk_inf_t;`
		
		`typedef struct {`
		`    m3_u32                dpc;`
		`    m3_u16                num_si;`
		`    m3_u8`
		`si_list[M3_MAX_SI_PER_RK];`
		`    m3_u16                num_opc;`
		`    m3_u32`
		`opc_list[M3_MAX_OPC_PER_RK];`
		`    m3_u16                num_ckt_range;`
		`    m3_ckt_range_t`
		`ckt_range[M3_MAX_CKT_RANGE_PER_RK];`
		`} m3_rk_elements_t;`

	as_id	Identifier of AS on which operation is to be performed. This is an unsigned integer value. 16 Most Significant Bits of this parameter should always be set to 0. The 16 least

		significant bits of this parameter can assume any value between 0 and (M3_MAX_AS – 1). M3_MAX_AS is defined in include file "m3ua_defines.h".
	rtctx	Routing context associated with the AS. It is mainly used to identify the traffic related to this AS. It is an unsigned integer value and can take values between 0x0 to 0xFFFFFFFE. It is primarily useful when a single SCTP association (ASP to SGP/ASP connection) is carrying traffic for more than one AS. Although this parameter is always configured but it may be avoided in M3UA messages by using API M3UA_CONN_OPT.
	rkey	Routing Key associated with the AS is specified in this parameter. It mainly consists of SS7 parameters using which an ASP can determine the AS for which traffic belongs. This parameter is utilized only when DATA message does not contain routing context. A very simple routing key would specify only SS7 service indicator (ISUP 0x05 & SCCP 0x03).
	trfmode	Mode of traffic handling applicable for the AS. AS can handle traffic in load-share/ broadcast/ over-ride mode.
	num_rtparam	Must be set to 1
	rtparam[]	Groups routing key parameters together
	dpc	This parameter should be set equal to DPC field of the incoming message that should be delivered to this AS.
	nw_app	Network Appearance associated with the traffic processed by this AS. AS routing key will be independent of Network Appearance if value of this parameter is set to M3_MAX_U32.
	num_si	Number of valid service indicators present in the parameter "si_list". This parameter can be set to 0.
	si_list[]	This parameter specifies the list of valid service indicators, which should be present in the incoming message that can be processed by this AS.
	num_opc	Number of valid Originating Point Codes present in the parameter "opc_list". This parameter can be set to 0.
	opc_list[]	This parameter specifies the list of valid Originating Point Codes, which should be present in the incoming message that can be processed by this AS.
	num_ckt_range	Number of valid Circuit Ranges present in the parameter "ckt_range". This parameter can be set to 0.
	ckt_range[]	This parameter specifies the list of valid combination of OPC and range of CIC values, which should be present in the incoming message that can be processed by this AS.
	confname	Name of AS configuration to be modified. Following are the configurations that may be modified: • M3_AS_RTCTX – Routing context of AS • M3_AS_RKEY – Routing Key of AS

		• M3_AS_INFO – Change all the configuration information associated with the AS
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get/modify AS configuration in the M3UA. This AS is responsible for processing the traffic received by the serving ASP with matching routing key parameters.	
Restrictions/Bugs	Routing Key should be added carefully as overlapping routing keys for more than one AS can result in erroneous behavior.	
Pre-requisites	M3UA_INIT, M3UA_NWAPP	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_as	

11.5 M3UA_R_ASP

Prototype	m3_s32 m3ua_r_asp(m3_u32 asp_id, m3_u8 oprn, m3ua_r_asp_t *pInf);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_r_asp_conf_t info;`
		`    } add;`
		`    struct {`
		`        m3_r_asp_conf_t info;`
		`    } get;`
		`} m3ua_r_asp_t;`
		
		`typedef struct`
		`{`
		`    m3_u32                **sctp_ep_id**;`
		`} m3_r_asp_conf_t;`

	asp_id	Identifier of Remote ASP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0. The 16 least significant bits of this parameter can assume any value between 0 and (M3_MAX_R_ASP – 1). The value of M3_MAX_R_ASP is defined in include file "m3ua_defines.h" and may be adjusted as per application requirements.
	sctp_ep_id	Remote SCTP Endpoint identifier for the Remote ASP. This is an unsigned integer value. This is a logical context identifier mainly given for the convenience of application developer. Internally M3UA library does not use this value.
Operations	M3_ADD, M3_DELETE, M3_GET	
Description	This API is used to add/delete/get Remote ASP configuration in the M3UA. This ASP is responsible to receive traffic for the remote AS it serves. This remote ASP should be brought into ACTIVE state before it starts receiving any traffic.	

Restrictions/Bugs	None
Pre-requisites	M3UA_INIT
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_r_asp

11.6 M3UA_R_AS

Prototype	m3_s32 m3ua_r_as(m3_u32 ras, m3_u8 oprn, m3ua_r_as_t *pparam);	
Parameters	oprn	Operation to be performed using this API.
	Associated C Data Structures	

```
typedef union {
    struct {
        m3_r_as_conf_t info;
    } add;
    struct {
        m3_r_as_conf_t info;
    } get;
    struct {
        m3_r_as_confname_t    confname;
        m3_r_as_conf_t        info;
    } modify;
} m3ua_r_as_t;

typedef struct
{
    m3_u32                rtctx;
    m3_rk_inf_t           rkey;
    m3_u8                 min_act_asp;
} m3_r_as_conf_t;

typedef struct
{
    m3_traffic_mode_t     trfmode;
    m3_u32                nw_app;
    m3_u8                 num_rtparam;
    m3_rk_elements_t
rtparam[M3_MAX_DPC_PER_RK];
} m3_rk_inf_t;

typedef struct {
    m3_u32                dpc;
    m3_u16                num_si;
    m3_u8
si_list[M3_MAX_SI_PER_RK];
    m3_u16                num_opc;
    m3_u32
opc_list[M3_MAX_OPC_PER_RK];
    m3_u16                num_ckt_range;
    m3_ckt_range_t
ckt_range[M3_MAX_CKT_RANGE_PER_RK];
} m3_rk_elements_t;

typedef enum {
```

		`M3_R_AS_RTCTX,`
		`M3_R_AS_RKEY,`
		`M3_R_AS_MIN_ACT_ASP,`
		`M3_R_AS_INFO`
		`}m3_r_as_confname_t;`

	ras	Identifier of AS on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0. The 16 least significant bits of this parameter can assume any value between 0 and (M3_MAX_R_AS – 1). The value of M3_MAX_R_AS is defined in include file “m3ua_defines.h”. This value may be adjusted as per application requirements.
	rtctx	Routing context associated with the remote AS. It is mainly used to identify the traffic related to this AS. It is useful mainly when a single SCTP association is carrying traffic for more than one AS. In simpler configurations this parameter may be avoided from M3UA messages using M3UA_CONN_OPT API.
	rkey	Routing Key associated with the remote AS is specified in this parameter. Based on this parameter outgoing messages are routed to the correct remote AS. Routing Key consists of 1 or more sets of SS7 parameters that would help in routing a SS7 message passed to M3UA from upper layers to the correct remote ASP that is serving this remote AS.
	num_rtparam	Must be set to 1
	rtparam[]	A set of SS7 routing parameters. This set of parameters consists of a DPC, list of service Indicators, list of originating point codes & list ISUP circuit ranges.
	trfmode	Mode of traffic handling applicable for the AS. AS can handle traffic in load-share/ broadcast/ over-ride mode. If mode is load-share, traffic is distributed dynamically based on SLS among the ACTIVE remote ASP serving this AS. In simpler configurations this parameter may be avoided from M3UA messages using M3UA_CONN_OPT API.
	dpc	This parameter should be set equal to DPC field of the outgoing message that should be sent to this AS.
	nw_app	Network Appearance associated with the traffic processed by this AS. The routing key will be independent of Network Appearance if value of this parameter is set to M3_MAX_U32.
	num_si	Number of valid service indicators present in the parameter “si_list”. This parameter can be set to 0.
	si_list[]	This parameter specifies the list of service indicators (SS7 user parts) for which this remote AS is handling traffic.
	num_opc	Number of valid Originating Point Codes present in the parameter “opc_list”. This parameter can be set to 0.
	opc_list[]	This parameter specifies the list of valid Originating Point Codes, which should be present in the outgoing DATA message that can be processed by this AS.
	num_ckt_range	Number of valid Circuit Ranges present in the parameter “ckt_range”. This parameter can be set to 0.

	ckt_range[]	This parameter specifies the list of valid combination of OPC and range of CIC values, which should be present in the outgoing DATA message that can be processed by the remote AS.
	min_act_asp	Minimum number of ASP that should be actively serving the AS if mode of traffic processing is load-share/broadcast.
	confname	Name of remote AS configuration to be modified. Following remote AS configurations may be modified: • M3_R_AS_RTCTX – Modify routing context of remote AS • M3_R_AS_RKEY – Modify routing key of remote AS • M3_R_AS_MIN_ACT_ASP – Modify minimum no. of active ASP requirement for the remote AS • M3_R_AS_INFO – Modify complete configuration information for the remote AS
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get/modify remote AS configuration in the M3UA. This remote AS handles all the traffic associated with its routing key.	
Restrictions/ Bugs	Routing Key should be added carefully as overlapping routing keys for more than one remote AS can result in erroneous behavior.	
Pre-requisites	M3UA_INIT, M3UA_NWAPP	
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_r_as	

11.7 M3UA_SGP

Prototype	m3_s32 m3ua_sgp(m3_u32 sgp, m3_u8 oprn, m3ua_sgp_t *pparam);	
Parameters	oprn	Operation to be performed using this API.
	Associated C Data Structures	(see below)

```
typedef union {
    struct {
        m3_sgp_conf_t info;
    } add;
    struct {
        m3_sgp_conf_t info;
    } get;
    struct {
        m3_sgp_confname_t    confname;
        m3_sgp_conf_t        info;
        m3_u32               ras_id;
        m3_u32               rasp_id;
    } modify;
} m3ua_sgp_t;

typedef struct
{
    m3_u32               sctp_ep_id;
    m3_u32               def_nwapp;
    struct {
        m3_bool_t        asp_list[M3_MAX_R_ASP];
```

		`    } `**`r_as_inf`**`[M3_MAX_R_AS];`
		`} m3_sgp_conf_t;`
		
		`typedef enum {`
		`    M3_SGP_NWAPP,`
		`    M3_SGP_ADD_R_ASP,`
		`    M3_SGP_DEL_R_ASP`
		`} m3_sgp_confname_t;`

	sgp	Identifier of SGP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0001. The least significant 16 bits of this parameter can assume any value between 0 and (M3_MAX_SGP – 1). For example SGP with identifier 1 must be encoded as 0x00010001. The value of M3_MAX_SGP is defined in include file “m3ua_defines.h” and may be adjusted as per application requirements.
	sctp_ep_id	SCTP Endpoint identifier for the SGP. It is an unsigned integer value. As every SGP is associated with an SCTP endpoint (combination of IP Address/SCTP Port). This value is just for convenience of application designer to associate an SCTP end point context with an SGP.
	def_nwapp	Default network appearance to be assumed while processing a message received without a network appearance at the SGP. This is an unsigned integer value. The network appearance specified in this parameter must be configured through M3UA_NWAPP API.
	r_as_inf	Remote Application Server related information specific to this SGP.
	asp_list	List of remote ASP serving the remote AS. For example, if remote application server process “rasp” is serving remote application server “ras” then r_as_inf[ras].asp_list[rasp] should be set to M3_TRUE else it should be set to M3_FALSE.
	confname	Name of SGP configuration to be modified. The values for this parameter are: • M3_SGP_NWAPP – Modify default network appearance • M3_SGP_ADD_R_ASP – Add a remote ASP to this SGP • M3_SGP_DEL_R_ASP – Delete a remote ASP from this SGP
	ras_id	Used to modify SGP configuration when confname parameter is set to M3_ASP_ADD_R_ASP/M3_ASP_DEL_R_ASP. A remote AS should be specified while adding/deleting a remote ASP to the SGP configuration.
	rasp_id	Used to modify SGP configuration when confname parameter is set to M3_ASP_ADD_R_ASP/ M3_ASP_DEL_R_ASP. The remote ASP specified to be added/ deleted is specified by this parameter. This remote ASP is serving remote AS specified by the parameter “ras_id”.
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get/modify SGP configuration in the M3UA. This SGP inter-works SS7 network(s) with the packet network.	

Restrictions/ Bugs	All the remote ASP and remote AS that are part of the SGP configuration should be added before adding the SGP. All connections from this SGP must be deleted before deleting this SGP.
Pre-requisites	M3UA_INIT, M3UA_NWAPP, M3UA_R_ASP, M3UA_R_AS
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_sgp

11.8 M3UA_R_SGP

Prototype	m3_s32 m3ua_r_sgp(m3_u32 rsgp, m3_u8 oprn, m3ua_r_sgp_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_r_sgp_conf_t info;`
		`    } add;`
		`    struct {`
		`        m3_r_sgp_conf_t info;`
		`    } get;`
		`} m3ua_r_sgp_t;`
		
		`typedef struct`
		`{`
		`    m3_u32              `**`sctp_ep_id`**`;`
		`} m3_r_sgp_conf_t;`

	rsgp	Identifier of Remote SGP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0001. The 16 least significant bits of this parameter can assume any value between 0 and (M3_MAX_R_SGP – 1). For example remote SGP with identifier 1 must be encoded as 0x00010001.
	sctp_ep_id	Remote SCTP Endpoint identifier for the Remote SGP. This value is not used by M3UA library internally. It is provided for convenience of application designer.
Operations	M3_ADD, M3_DELETE, M3_GET	
Description	This API is used to add/delete/get Remote SGP configuration in the M3UA. This SGP is responsible to receive traffic for the SG it serves. It is necessary to associate an SGP with an SG, so that it starts receiving traffic.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_r_sgp	

11.9 M3UA_SG

Prototype	m3_s32 m3ua_sg(m3_u32 sg, m3_u8 oprn, m3ua_sg_t *pparam);	
Parameters	oprn	Operation to be performed using this API.
	Associated C Data Structures	

```
typedef union {
    struct {
        m3_sg_conf_t info;
    } add;
    struct {
        m3_sg_conf_t info;
    } get;
    struct {
        m3_sg_confname_t    confname;
        m3_sg_conf_t        info;
    } modify;
} m3ua_sg_t;

typedef struct
{
    m3_sg_mode_t          sgmode;
    m3_u8                 num_sgp;
    m3_u32
sgp_list[M3_MAX_SGP_PER_SG];
    m3_u8                 rt_per_rc_supp;
} m3_sg_conf_t;

typedef enum {
    M3_SG_MODE,
    M3_SG_SGP_LIST,
    M3_SG_INFO
} m3_sg_confname_t;
```

	sg	Identifier of SG on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0001. The 16 least significant bits of this parameter can assume any value between 0 and (M3_MAX_SG – 1). For example SG with identifier 1 must be encoded as 0x00010001.
	sgmode	Mode of traffic handling applicable for the Signaling Gateway. SG can handle traffic in load-sharing/broadcast mode. The values of this parameter can be: M3_SGMODE_LOADSHARE (0x01) – Load sharing mode M3_SGMODE_BROADCAST (0x02) – Broadcast mode
	num_sgp	This parameter specifies number of remote SGP that are responsible for handling traffic for this SG.
	sgp_list[]	This parameter specifies list of remote SGP that are responsible for handling traffic for this SG.
	rt_per_rc_supp	This parameter specifies if SG is capable of handling route states at per routing context level. If a single SCTP association is carrying traffic for more than one routing context, then SG is able to control route states for traffic flow belonging to each

		individual routing context. This parameter is applicable only when ROUTE_PER_RC feature is enabled. Once this feature is enabled, it is expected that application on SG would provide list of routing contexts to M3UA_PAUSE, M3UA_RESUME and M3UA_STATUS primitives.
	confname	Name of SG configuration to be modified. Following configurations may be modified for the SG: • M3_SG_MODE – Modify traffic handling mode of the SG • M3_SG_SGP_LIST – Modify list of remote SGP's handling traffic for this SG • M3_SG_INFO – Modify complete configuration for this SG
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get SG configuration in the M3UA. This SG is responsible for inter-working SS7 network(s) with the packet network.	
Restrictions/ Bugs	All remote SGP that are serving this SG must be deleted before deleting this SG.	
Pre-requisites	M3UA_INIT, M3UA_R_SGP	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_sg	

11.10 M3UA_ASP_STATE

Prototype	m3_s32 m3ua_asp_state(m3_u32 asp, m3_u32 rsp, m3_u8 oprn, m3ua_asp_state_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        struct {`
		`            m3_u32        `**`as_id`**`;`
		`        } info;`
		`        m3_asp_state_t    `**`state`**`;`
		`    } get;`
		`    struct {`
		`        struct {`
		`            m3_u16        `**`num_as`**`;`
		`            m3_u32        as_list[M3_MAX_AS];`
		`        }info;`
		`        m3_asp_state_t    `**`state`**`;`
		`    } modify;`
		`} m3ua_asp_state_t;`

	asp	Identifier of ASP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0000. This parameter can assume any value between 0 and (M3_MAX_ASP – 1). For example ASP with identifier 1 must be encoded as 0x00000001.

	rsp	Remote Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a remote SGP with identifier 2 while 0x00000003 identifies remote ASP with identifier 3.
	as_id	Application server identifier in which state of ASP is to modified or required.
	state	State of ASP. The state of ASP may be one of following: • M3_ASP_DOWN – ASP State DOWN • M3_ASP_INACTIVE – ASP State INACTIVE • M3_ASP_ACTIVE – ASP State ACTIVE
	num_as	Number of Application Servers present in the "as_list" parameter.
	as_list[]	List of Application Servers in which state of ASP is to be modified.
Operations	M3_GET, M3_MODIFY	
Description	This API is used to get/modify ASP state in the M3UA. This API sends ASPSM/ASPTM messages based on the state transition of the ASP.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_R_SGP, M3UA_ASP	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_asp_state	

11.11 M3UA_R_ASP_STATE

Prototype	m3_s32 m3ua_r_asp_state(m3_u32 rasp, m3_u32 lsp, m3_u8 oprn, m3ua_r_asp_state_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	
		```
		typedef union {
		    struct {
		        struct {
		            m3_u32            as_id;
		        } info;
		        m3_asp_state_t    state;
		    } get;
		} m3ua_r_asp_state_t;
		```

	rasp	Identifier of remote ASP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0000. This parameter can assume any value between 0 and (M3_MAX_R_ASP – 1). For example remote ASP with identifier 1 must be encoded as 0x00000001.
	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP

		with identifier 3.
	as_id	Remote Application server identifier in which state of ASP is required.
	state	State of remote ASP. The state of remote ASP may be one of following: • M3_ASP_DOWN – ASP State DOWN • M3_ASP_INACTIVE – ASP State INACTIVE • M3_ASP_ACTIVE – ASP State ACTIVE
Operations	M3_GET	
Description	This API is used to get remote ASP state in the M3UA.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP, M3UA_SGP	

11.12 M3UA_R_AS_STATE

Prototype	m3_s32 m3ua_r_as_state(m3_u32 ras, m3_u32 lsp, m3_u8 oprn, m3ua_r_as_state_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_as_state_t    state;`
		`    } get;`
		`} m3ua_r_as_state_t;`

	ras	Identifier of remote AS on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0000. This parameter can assume any value between 0 and (M3_MAX_R_AS – 1). For example remote AS with identifier 1 must be encoded as 0x00000001.
	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	state	State of remote AS. The state of remote AS may be one of the following: • M3_AS_DOWN – Remote AS is down • M3_AS_INACTIVE – Remote AS is inactive • M3_AS_ACTIVE – Remote AS is active • M3_AS_PENDING – Remote AS is in pending state (it is transitioning from active to inactive state)
Operations	M3_GET	
Description	This API is used to get remote AS state in the M3UA. State of remote AS depends on the combined state of remote ASP serving this AS.	
Restrictions/Bugs	None	

Pre-requisites	M3UA_INIT, M3UA_ASP, M3UA_SGP

11.13 M3UA_CONN

Prototype	m3_s32 m3ua_conn(m3_u32 lsp, m3_u32 rsp, m3_u8 oprn, m3ua_conn_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	typedef union {
		    struct {
		        m3_conn_conf_t info;
		    } add;
		    struct {
		        m3_conn_conf_t info;
		    } get;
		    struct {
		        m3_conn_confname_t **confname**;
		        m3_conn_conf_t info;
		    } modify;
		} m3ua_conn_t;
		
		typedef struct
		{
		    m3_u32 **assoc_id**;
		    m3_u32 **i_str**;
		    m3_u32 **o_str**;
		} m3_conn_conf_t;
		
		typedef enum {
		    M3_CONN_ASSOC,
		    M3_CONN_I_STR,
		    M3_CONN_O_STR
		} m3_conn_confname_t;

	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	rsp	Remote Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between remote ASP and remote SGP. For example 0x00010002 identifies a remote SGP with identifier 2 while 0x00000003 identifies remote ASP with identifier 3.
	assoc_id	SCTP Association identifier mapped to this M3UA connection.
	i_str	Number of in streams present in the association.
	o_str	Number of out streams present in the association.
	confname	Name of Connection configuration to be modified.
Operations	M3_ADD, M3_DELETE, M3_GET, M3_MODIFY	
Description	This API is used to add/delete/get/modify connection configuration in	

	M3UA. A connection is added with connection state as M3_CONN_NOT_ESTB. M3UA_CONN_STATE API should be invoked after this API to change the connection state to M3_CONN_ESTB.
Restrictions/Bugs	None
Pre-requisites	M3UA_INIT, M3UA_ASP, M3UA_SGP, M3UA_R_ASP, M3UA_R_SGP
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_conn

11.14 M3UA_CONN_STATE

Prototype	m3_s32 m3ua_conn_state(m3_u32 lsp, m3_u32 rsp, m3_u8 oprn, m3ua_conn_state_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_conn_state_t` **`state`**`;`
		`    } get;`
		`    struct {`
		`        m3_conn_state_t` **`state`**`;`
		`    } modify;`
		`} m3ua_conn_state_t;`
		
		`typedef enum`
		`{`
		`    M3_CONN_NOT_ESTB       = 0,`
		`    M3_CONN_SETUP_IN_PROG = 1,`
		`    M3_CONN_ESTB           = 2,`
		`    M3_CONN_CONG_1         = 3,`
		`    M3_CONN_CONG_2         = 4,`
		`    M3_CONN_CONG_3         = 5,`
		`    M3_CONN_ALIVE          = 6`
		`} m3_conn_state_t;`

	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	rsp	Remote Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between remote ASP and remote SGP. For example 0x00010002 identifies a remote SGP with identifier 2 while 0x00000003 identifies remote ASP with identifier 3.
	state	State of Connection. Following are connection states: • M3_CONN_NOT_ESTB – Connection is not established • M3_CONN_SETUP_IN_PROG – Connection setup is in progress • M3_CONN_ESTB – Connection is established

		• M3_CONN_CONG_1 – Congestion Level 1 • M3_CONN_CONG_2 – Congestion Level 2 • M3_CONN_CONG_3 – Congestion Level 3
Operations	M3_GET, M3_MODIFY	
Description	This API is used to get/modify connection state in M3UA. A connection state must be modified by the layer management entity whenever it receives any indication from SCTP related to association mapped to this M3UA connection.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_CONN	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_conn_state	

11.15 M3UA_CONN_OPT

Prototype	m3_s32 m3ua_conn_opt(m3_u32 lsp, m3_u32 rsp, m3_u8 oprn, m3ua_conn_opt_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_conn_opt_t **opt**;`
		`    } get;`
		`    struct {`
		`        m3_conn_opt_t **opt**;`
		`    } modify;`
		`} m3ua_conn_opt_t;`

	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	rsp	Remote Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between remote ASP and remote SGP. For example 0x00010002 identifies a remote SGP with identifier 2 while 0x00000003 identifies remote ASP with identifier 3.
	opt	Options to be enabled for this connection. By default all the options are switched OFF. This is an unsigned integer value. Following are the values for this field: • No Options Enabled (Default) 0x00000000 • Exclude Routing Context 0x00000001 • Exclude Traffic Mode 0x00000002 • Exclude Network Appearance 0x00000004 • Exclude ASP ID 0x00000008

		One or more of above values may be ORed together to set multiple options together. For example 0x0000000F would mean that routing context, traffic mode, ASPID & network appearance should be excluded.
Operations	M3_GET, M3_MODIFY	
Description	This API is used to get/modify connection options in M3UA. A connection option may be modified by the layer management entity whenever required. It is recommended that exclusion of parameters is done only in case of simple configurations.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_CONN	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_conn_opt	

11.16 M3UA_ROUTE

Prototype	m3_s32 m3ua_route(m3_u8 oprn, m3ua_route_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	
		```
		typedef union {
		    struct {
		        m3_rt_conf_t       info;
		    } add;
		    struct {
		        m3_rt_conf_t       info;
		    } del;
		} m3ua_route_t;

		typedef struct
		{
		    m3_pc_inf_t             pc_inf;
		    m3_u32                  le_id;
		    m3_u8                   priority;
		} m3_rt_conf_t;
		```

	pc_inf	Point code related information towards which the route is being added or deleted. This parameter identifies a network and a point code in that network.
	le_id	Signaling Gateway ID through which traffic for the specified SS7 point code can be routed.
	priority	Priority associated with this route. A higher value means a higher priority.
Operations	M3_ADD, M3_DELETE	
Description	This API is used to add/delete routes to SS7 point codes. A SS7 network can be reached through a Signaling Gateway. A high priority available route is always chosen before a low priority route while routing traffic. Route state while initial addition is set to M3_RTSTATE_DOWN. Route state gets modified after receiving SSNM messages from remote SGP.	

Restrictions/Bugs	Same route should not be added more than one time as no checks are put on the parameters of the API.
Pre-requisites	M3UA_INIT, M3UA_NWAPP, M3UA_SG
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_route

11.17 M3UA_ROUTE_FROM_OPC

Prototype	m3_s32 m3ua_route_from_opc(m3_u8 oprn, m3_u32 opc, m3ua_route_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_rt_conf_t       info;`
		`    } add;`
		`    struct {`
		`        m3_rt_conf_t       info;`
		`    } del;`
		`} m3ua_route_t;`
		
		`typedef struct`
		`{`
		`    m3_pc_inf_t           `**`pc_inf`**`;`
		`    m3_u32                `**`le_id`**`;`
		`    m3_u8                 `**`priority`**`;`
		`} m3_rt_conf_t;`

	opc	This route specifically belongs to traffic originating from this point code. It means that for traffic originating from point code “opc” and destined for point code specified in “pc_inf.pc” would be sent through this route if it is available.
	pc_inf	Point code related information towards which the route is being added or deleted. This parameter identifies a network and a point code in that network.
	le_id	Signaling Gateway ID through which traffic for the specified SS7 point code can be routed.
	priority	Priority associated with this route. A higher value means a higher priority.
Operations	M3_ADD, M3_DELETE	
Description	This API is used to add/delete routes to SS7 point codes. A SS7 network can be reached through a Signaling Gateway. A high priority available route is always chosen before a low priority route while routing traffic. Route state while initial addition is set to M3_RTSTATE_DOWN. Route state gets modified after receiving SSNM messages from remote SGP.	
Restrictions/Bugs	Same route should not be added more than one time as no checks are put on the parameters of the API.	
Pre-requisites	M3UA_INIT, M3UA_NWAPP, M3UA_SG	

Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_route

11.18 M3UA_USER

Prototype	m3_s32 m3ua_user(m3_u16 user, m3_u8 oprn, m3ua_route_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_usr_conf_t    info;`
		`    } add;`
		`} m3ua_user_t;`
		
		`typedef struct`
		`{`
		`    m3_u32                 `**`sp_id`**`;`
		`    union {`
		`        struct {`
		`            m3_u8          `**`sio`**`;`
		`            m3_u32         `**`as_id`**`;`
		`        } mtp_user;`
		`        struct {`
		`            m3_bool_t      `**`a_data`**`;`
		`        } nif_user;`
		`    } user;`
		`} m3_usr_conf_t;`

	user	Identifies User on which operation is to be performed. It can assume any value between 0 and (M3_MAX_USR – 1).
	sp_id	Local Signaling Point identifier through which user shall send/receive the traffic. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	sio	Service Indicator specifying the SS7 user part.
	as_id	Application Server for which this user is responsible to send/receive traffic.
	a_data	This parameter must always be set to M3_TRUE when "sp_id" parameter specifies that user belongs to SGP.
Operations	M3_ADD, M3_DELETE	
Description	This API is used to add/delete SS7 User parts which responsible for sending/receiving SS7 traffic. A User is associated with a local Application server that in turn identifies a particular SS7 traffic flow in M3UA.	
Restrictions/Bugs	Multiple users with same configuration should not be added to M3UA.	
Pre-requisites	M3UA_INIT, M3UA_ASP, M3UA_SGP, M3UA_AS	

Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_user

11.19 M3UA_TIMER

Prototype	m3_s32 m3ua_timer(m3_u8 oprn, m3ua_timer_t *pparam);	
Parameters	oprn	Operation to be performed using this API.

	Associated C Data Structures	typedef struct {
		    m3_timer_type_t **type**;
		    union {
		        m3_aspmtimer_t aspmtimer;
		        m3_pdtimer_t pdtimer;
		        m3_hbeattimer_t hbeattimer;
		        m3_rkmtimer_t rkmtimer;
		    } get;
		    union {
		        m3_aspmtimer_t aspmtimer;
		        m3_pdtimer_t pdtimer;
		        m3_hbeattimer_t hbeattimer;
		        m3_rkmtimer_t rkmtimer;
		    } modify;
		} m3ua_timer_t;
		
		typedef enum {
		    M3_TIMER_TYPE_ASPM,
		    M3_TIMER_TYPE_PD,
		    M3_TIMER_TYPE_HBEAT,
		    M3_TIMER_TYPE_RKM
		} m3_timer_type_t;
		
		typedef struct {
		    m3_u16 **retry**;
		    m3_u16 **sw_try**;
		    m3_u16 **l_dur**;
		    m3_u16 **h_dur**;
		} m3_aspmtimer_t;
		
		typedef struct {
		    m3_u16 **dur**;
		} m3_pdtimer_t;
		
		typedef struct {
		    m3_u16 **retry**;
		    m3_u16 **dur**;
		} m3_hbeattimer_t;
		
		typedef struct {
		    m3_u16 **retry**;
		    m3_u16 **dur**;
		} m3_rkmtimer_t;

	type	Specifies the timer type on which operation is to be

		performed.
	retry	Number of times ASPSM/ASPTM/RKM message should be re-sent.
	sw_try	Number of times ASPSM/ASPTM message should be repeated with “l_dur” duration. After this many retries the duration between repetitions is changed to “h_dur” duration.
	l_dur	Duration between fast repetitions.
	h_dur	Duration between slow repetitions.
	dur	Duration for AS-PENDING timer/heartbeat timer or RKM timer.
Operations	M3_GET, M3_MODIFY	
Description	This API is used to configure timer durations and number of retries associated with ASPM procedures. AS-Pending timer duration and heartbeat timer duration are also adjusted using this API. All the timer durations should be provided in terms of M3_TICKS_PER_SEC ticks. For example: if M3_TICKS_PER_SEC is 100, then 2 seconds would be configured as 200.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_timer	

11.20 M3UA_HEARTBEAT

Prototype	m3_s32 m3ua_heartbeat(m3_u32 lsp, m3_u32 rsp, m3_u8 oprn, void *pparam);	
Parameters	oprn	Operation to be performed using this API.
	Associated C Data Structures	
	lsp	Local Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between ASP and SGP. For example 0x00010002 identifies a SGP with identifier 2 while 0x00000003 identifies ASP with identifier 3.
	rsp	Remote Signaling Point identifier. Based on most significant 16 bits this parameter is used to distinguish between remote ASP and remote SGP. For example 0x00010002 identifies a remote SGP with identifier 2 while 0x00000003 identifies remote ASP with identifier 3.
Operations	M3_ADD, M3_DELETE	
Description	This API is used to start/stop (add/delete) heartbeat procedure on the connection identified by the lsp-rsp combination. This procedure is required only when the underlying transport layer does not support any type of auditing for the association between the two endpoints.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_CONN_STATE	

Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_heartbeat

11.21 M3UA_R_ASPLOCK

Prototype	m3_s32 m3ua_r_asplock(m3_u32 rasp, m3_u8 oprn, void *pparam);	
Parameters	oprn	Operation to be performed using this API.
	rasp	Identifier of remote ASP on which operation is to be performed. 16 Most Significant Bits of this parameter should always be set to 0x0000. 16 least significant bits of this parameter can assume any value between 0 and (M3_MAX_R_ASP – 1). For example remote ASP with identifier 1 must be encoded as 0x00000001.
Operations	M3_ADD, M3_DELETE	
Description	This API is used to add/delete lock on a remote ASP. If a remote ASP is locked in the present state. It can move to DOWN state from ACTIVE/INACTIVE state and INACTIVE state from ACTIVE state.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_R_ASP	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_r_asplock	

11.22 M3UA_RKEY

Prototype	m3_s32 m3ua_rkey(m3_u8 oprn, m3ua_rkey_t *pparam);	
Parameters	oprn	Operation to be performed using this API.
	Associated C Data Structures	typedef union { struct { m3_u32 asp_id; m3_u32 rsp_id; m3_u16 num_as; m3_u32 as_list[M3_MAX_AS]; } reg; struct { m3_u32 asp_id; m3_u32 rsp_id; m3_u16 num_as; m3_u32 as_list[M3_MAX_AS]; } dereg; struct { m3_u32 lsp_id; m3_u32 rasp_id; m3_u16 num_result; struct { m3_u32 as_id; m3_u32 reg_status; } result[M3_MAX_RK]; } status; } m3ua_rkey_t;

```
typedef union {
    struct {
        m3_u32              asp_id;
        m3_u32              rsp_id;
        m3_u16              num_as;
        m3_u32              as_list[M3_MAX_AS];
    } reg;
    struct {
        m3_u32              asp_id;
        m3_u32              rsp_id;
        m3_u16              num_as;
        m3_u32              as_list[M3_MAX_AS];
    } dereg;
    struct {
        m3_u32              lsp_id;
        m3_u32              rasp_id;
        m3_u16              num_result;
        struct {
            m3_u32          as_id;
            m3_u32          reg_status;
        } result[M3_MAX_RK];
    } status;
} m3ua_rkey_t;
```

	M3_REGISTER	
	asp_id	ASP ID initiating RK Registration procedure. This ASP has already been provisioned in the M3UA library.
	rsp_id	Identifier of remote Signaling point (remote SGP or remote IPSP) to which routing key Registration Request is to be sent.
	num_as	Number of valid elements in the as_list array.
	as_list[]	List of AS identifiers for which routing key registration has been requested. These Application Servers are already provisioned in the M3UA library.
	M3_DEREGISTER	
	asp_id	Identifier of the ASP initiating RK De-registration procedure. This ASP has already been provisioned in the M3UA library.
	rsp_id	Identifier of remote Signaling point (remote SGP or remote IPSP) to which routing key De-registration Request is to be sent.
	num_as	Number of valid elements in the as_list array.
	as_list[]	List of AS identifiers for which routing key registration/de-registration has been requested. These Application Servers are already provisioned in the M3UA library.
	M3_STATUS	
	lsp_id	Local Signaling point identifier (SGP or ASP) authorizing the Registration of a Routing Key. When a RK registration request is received from remote ASP, it is passed to the local ASP/SGP for authorization through M3UA_MGMT_NTFY API. Once local ASP/SGP authorizes (using M3_STATUS operation) the Routing Key, it will use this API to complete the RK registration procedure.
	rasp_id	Remote ASP ID that requested the Registration of a routing key.
	num_result	Number of Registration results present in parameter result.
	result[]	Registration / De-Registration Results
	as_id	Remote AS ID whose registration status is confirmed to the local Signaling point.
	reg_status	Registration status of the remote AS
Operations	M3_REGISTER, M3_DEREGISTER, M3_STATUS	
Description	This API is used to Register/Deregister an Application Server Routing Key in the remote Signaling point (IPSP or SGP). This is also used by the Layer management to authorize registration of the remote AS.	
Restrictions/ Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP, M3UA_AS, M3UA_CONN	

11.23 M3UA_SET_TRACEMAP

Prototype	void m3ua_set_trace_map(m3_u32 aTraceMap);	
Parameters	aTraceMap	Trace levels that are to be switched ON in the M3UA library. Following are the trace levels:

		• m3uaErrorTrace – Error traces • m3uaConfigTrace – Configuration traces • m3uaAspmTrace – ASPM traces • m3uaRkmTrace – RKM traces • m3uaSsnmTrace – SSNM traces • m3uaTxrTrace – Transfer traces • m3uaMgmtTrace – Management traces • m3uaInMsgTrace – In Message traces • m3uaOutMsgTrace – Out Message traces • m3uaStartupTrace – Startup traces • m3uaTimerTrace – Timer traces More than one trace level may be switched ON by OR'ing above trace levels. For example to switch ON error and timer traces, following may be used: `aTraceMap = m3uaErrorTrace | m3uaTimerTrace;`
Operations		
Description	This API is used to switch ON/OFF the logging from M3UA library.	
Restrictions/Bugs	Presently, tracing support is not complete. Logs are still being built in the M3UA library.	
Pre-requisites		

11.24 M3UA_DIAG

Prototype	void m3ua_diag (m3_u32 diagType, m3_s8 *diagStr, m3_u32 len);	
Parameters	diagType	This parameter specifies the type(s) of diagnostics to be collected from M3UA protocol library. The type of diagnostics available are: M3_MEM_DIAG – Memory related diagnostics M3_TIMER_DIAG – Timer related diagnostics M3_ALL_DIAG – All the above mentioned diagnostics
	diagStr	This is a pre-allocated character array (or string) that needs to be passed to the M3UA library. M3UA would store a printable string in this parameter. The layer management entity may print this string for visual inspection of the current status of memory/timer resources being used by M3UA library. The important point to note here is that the LM entity needs to allocate memory for this parameter. The recommended value is 4096 bytes.
	len	Length of the diagStr parameter. If 4096 bytes have been allocated for diagStr, then this parameter must be set to 4096.
Operations		

Description	This API is used to collect diagnostics information from M3UA library for visual inspection.
Restrictions/Bugs	None
Pre-requisites	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_diag

11.25 M3UA_PFM

Prototype	void m3ua_pfm (m3_u8 oprn, m3_u32 pfmType, m3ua_pfm_t *pInf);	
Parameters	oprn	Operations to be performed using this API.

	Associated C Data Structures	`typedef union {`
		`    struct {`
		`        m3_bool_t        resetKPI;`
		`        m3_pfm_data_t    kpiData;`
		`    } get;`
		`} m3ua_pfm_t;`
		
		`typedef struct {`
		`    m3_u32                aspmKPI[M3_PFM_ASPM_MAX];`
		`    m3_u32                txrKPI[M3_PFM_TXR_MAX];`
		`    m3_u32                mgmtKPI[M3_PFM_MGMT_MAX];`
		`    m3_u32                ssnmKPI[M3_PFM_SSNM_MAX];`
		`    m3_u32                rkmKPI[M3_PFM_RKM_MAX];`
		`    m3_u32               timerKPI[M3_PFM_TIMER_MAX];`
		`    m3_u32               thrptKPI[M3_PFM_THRPT_MAX];`
		`} m3_pfm_data_t;`

	pfmType	This parameter specifies the type of KPI for which collection is to be enabled or disabled. The value for this parameter may be: M3_PFM_CTG_ASPM -- KPI for ASPM Message counters M3_PFM_CTG_TXR -- KPI for TXR Message counters M3_PFM_CTG_MGMT -- KPI for MGMT Message counters M3_PFM_CTG_SSNM -- KPI for SSNM Message counters M3_PFM_CTG_RKM -- KPI for RKM Message counters M3_PFM_CTG_TIMER -- KPI for TIMER operation counters M3_PFM_CTG_THRPT -- KPI for throughput measurements M3_PFM_ALL -- All the above mentioned KPI
	resetKPI	This parameter commands M3UA to reset all the KPI counters to zero after reporting them in the "`kpiData`" parameter.
	kpiData	This parameter contains the counters for various KPI maintained by M3UA library. aspmKPI – ASPM message counters txrKPI – TRANSFER message counters mgmtKPI – MGMT message counters ssnmKPI – SSNM message counters rkmKPI – RKM message counters

		timerKPI – TIMER message counters thrptKPI – Throughput / bitrate counters Following is an example illustrating how KPI data must be read. To get the no. of ASPUP messages transferred by M3UA, “`aspmKPI[M3_PFM_ASPM_MSG_ASPUP_TX]`“ parameter must be read. Similarly to get the total no. of bytes received by M3UA, ”`thrptKPI[M3_PFM_THRPT_TOT_BYTES_RX]`“ parameter must be read.
Operations	M3_ADD, M3_DELETE, M3_GET	
Description	This API is used to collect Performance Data (or KPI – Key Performance Indicators) from M3UA library.	
Restrictions/Bugs	None	
Pre-requisites		
Code Sample	Please refer source file comm/src/common.c, function name um3_m3ua_pfm	

11.26 M3UA_FEATURE

Prototype	void m3ua_feature (m3_u8 oprn, m3_u32 feature_bitmap);	
Parameters	oprn	Operations to be performed using this API.
	Associated C Data Structures	
	feature_bitmap	This parameter specifies the bitmap of features that are to be enabled or disabled. Presently it is used to control only one feature: ROUTE-PER-LRC (0x00000001) – Point Code Routes (SS7 Routes) to be maintained on per Routing Context basis
Operations	M3_ADD, M3_DELETE	
Description	This API is used to enable or disable a M3UA feature.	
Restrictions/Bugs	None	
Pre-requisites		
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_feature	

12 Layer Management API (M3UA→LM)

12.1 M3UA_MGMT_NTFY

Prototype	m3_s32 m3ua_mgmt_ntfy(m3ua_mgmt_ntfy_t *);	
Parameters	Associated C Data Structures	

```
typedef struct {
    m3_u8             type;
    union {
        struct {
            m3_u32           lsp_id;
            m3_u32           rsp_id;
            m3_u32           err_code;
            m3_u16           num_rc;
            m3_u32
rc_list[M3_MAX_RTCTX];
            m3_u16           num_pc;
            m3_u32
pc_list[M3_MAX_PC_SSNM];
            m3_u32           nw_app;
            m3_u16           diag_len;
            m3_u8            *p_diag_inf;
        } err;
        struct {
            m3_u32           asp_id;
            m3_u32           rsp_id;
            m3_u32           as_id;
            m3_asp_state_t   state;
        } asp;
        struct {
            m3_u32           asp_id;
            m3_u32           lsp_id;
            m3_u32           as_id;
            m3_asp_state_t   state;
        } r_asp;
        struct {
            m3_u32           as_id;
            m3_u32           lsp_id;
            m3_u32           rsp_id;
            m3_as_state_t    state;
        } as;
        struct {
            m3_u32           as_id;
            m3_u32           lsp_id;
            m3_as_state_t    state;
        } r_as;
        struct {
            m3_u32           lsp_id;
            m3_u32           rsp_id;
            m3_conn_state_t  state;
        } conn;
        struct {
            m3_u32           lsp_id;
            m3_u32           rsp_id;
```

```
                m3_u16           status_type;
                m3_u16           status_inf;
                m3_u32           m3asp_id;
                m3_u16           num_as;
                m3_u32
as_list[M3_MAX_RTCTX];
            } notify;
            struct {
                m3_u32           lsp_id;
                m3_u32           asp_id;
                m3_u16           num_as;
                m3_u32           as_list[M3_MAX_RK];
            } reg;
            struct {
                m3_u32           lsp_id;
                m3_u32           asp_id;
                m3_u16           num_as;
                m3_u32           as_list[M3_MAX_RK];
            } dereg;
            struct {
                m3_u32           asp_id;
                m3_u32           rsp_id;
                m3_u16           num_as;
                struct {
                    m3_u32       as_id;
                    m3_u32       rtctx;
                    m3_u32       status;
                } result[M3_MAX_RK];
            } reg_status;
            struct {
                m3_u32           asp_id;
                m3_u32           rsp_id;
                m3_u16           num_as;
                struct {
                    m3_u32       as_id;
                    m3_u32       rtctx;
                    m3_u32       status;
                } result[M3_MAX_RK];
            } dreg_status;
        } param;
} m3ua_mgmt_ntfy_t;
```

	type	Type of management notification received from M3UA. This parameter is used to determine the valid parameters in the union. This is an unsigned 8 bit parameter with following values: • M3_MGMT_NTFY_ASP_STATE (0x01) • M3_MGMT_NTFY_AS_STATE (0x02) • M3_MGMT_NTFY_R_ASP_STATE (0x03) • M3_MGMT_NTFY_R_AS_STATE (0x04) • M3_MGMT_NTFY_CONN_STATE (0x05) • M3_MGMT_NTFY_NOTIFY (0x06) • M3_MGMT_NTFY_ERR (0x07) • M3_MGMT_NTFY_REGISTER (0x08)

		• M3_MGMT_NTFY_DEREGISTER (0x09) • M3_MGMT_NTFY_REG_STATUS (0x0a) • M3_MGMT_NTFY_DEREG_STATUS (0x0b)
	M3_MGMT_NTFY_ERR	
	err	Container for error notification parameters.
	lsp_id	Local signaling point identifier which has received M3UA ERROR message.
	rsp_id	Remote signaling point identifier that has sent M3UA ERROR message.
	err_code	Error code received in the M3UA ERROR message.
	num_rc	Number of Routing Contexts received in the M3UA ERROR message.
	rc_list	List of Routing Contexts received in the M3UA ERROR message.
	num_pc	Number of Point codes received in the M3UA ERROR message.
	pc_list	List of Point Codes received in the M3UA ERROR message.
	nw_app	Network Appearance received in the M3UA ERROR message.
	diag_len	Length of diagnostic message received in M3UA ERROR message.
	p_diag_inf	Pointer to diagnostic information received in the M3UA Error message.
	M3_MGMT_NTFY_ASP_STATE	
	asp	Container for ASP State notification parameters.
	asp_id	Application Server Process ID whose state has changed.
	as_id	Application Server ID for which ASP state has changed
	state	State of ASP
	rsp_id	Remote signaling point ID for which ASP state change has taken place
	M3_MGMT_NTFY_R_ASP_STATE	
	r_asp	Container for remote ASP State notification parameters.
	asp_id	Remote Application Server Process ID whose state has changed.
	as_id	Remote Application Server ID for which RASP state has changed.
	state	State of Remote ASP
	lsp_id	Local signaling point ID for which RASP state has changed
	M3_MGMT_NTFY_AS_STATE	
	as	Container for AS state notification parameters.
	as_id	Application Server ID whose state has changed
	lsp_id	Local signaling point ID in which AS state has changed
	rsp_id	Remote signaling point ID in which AS state has changed
	state	State of AS
	M3_MGMT_NTFY_R_AS_STATE	
	r_as	Container for remote AS state notification parameters.
	as_id	Remote Application Server ID whose state has changed

	lsp_id	Local signaling point ID in which RAS state has changed
	state	State of Remote AS.
	M3_MGMT_NTFY_CONN_STATE	
	conn	Container for connection state notification parameters.
	lsp_id	Local signaling point ID for which connection state has changed
	rsp_id	Remote signaling point ID for which connection state has changed
	state	State of Connection.
	M3_MGMT_NTFY_NOTIFY	
	notify	Container for Notify parameters received from remote signaling point.
	lsp_id	Local signaling point ID which has received M3UA NTFY message.
	rsp_id	Remote signaling point ID which has sent M3UA NTFY message.
	status_type	Status type parameter received in the M3UA NTFY message. This is encoded as per RFC 3332.
	status_inf	Status Information parameter received in the M3UA NTFY message. This is encoded as per RFC 3332.
	m3asp_id	M3UA ASP identifier received in the M3UA NTFY message.
	num_as	Number of AS associated with the notification.
	as_list[]	List of AS associated with the notification.
	M3_MGMT_NTFY_REGISTER	
	reg	Container for the Registration request parameters received from the remote ASP.
	lsp_id	Local signaling point identifier that has received M3UA REG REQ message.
	asp_id	Remote Application Server Process ID that has sent M3UA REG REQ message.
	num_as	Number of AS associated with the notification.
	as_list[]	List of AS associated with the notification.
	M3_MGMT_NTFY_DEREGISTER	
	dereg	Container for the De-Registration request parameters received from the remote signaling point.
	lsp_id	Local signaling point ID that has received M3UA DEREG REQ message
	asp_id	Remote Application Server Process ID that has sent M3UA DEREG REQ message.
	num_as	Number of AS present in the as_list array.
	as_list[]	List of AS that have been deregistered.
	M3_MGMT_NTFY_REG_STATUS	
	reg_status	Container for the Registration result parameters received in the registration result from the remote Signaling point.
	asp_id	Application Server Process ID for which M3UA REG RSP message is received
	rsp_id	Remote signaling point ID that has sent M3UA REG RSP

		message.
	num_as	Number of AS present in the result array.
	result[]	Result for the M3UA RK registration procedure.
	as_id	Application Server ID for which Registration result is received
	rtctx	Routing Context of the Application Server
	status	Registration status of the Application Server. The values for this parameter are same as “Registration Status” parameter defined in RFC 3332.
	M3_MGMT_NTFY_DEREG_STATUS	
	dreg_status	Container for the De-Registration result parameters received in the De-Registration result from the remote Signaling point.
	asp_id	Application Server Process ID for which M3UA DEREG RSP message is received
	rsp_id	Remote signaling point ID that has sent M3UA DEREG RSP message.
	num_as	Number of AS present in the result array.
	result[]	Result for the M3UA RK deregistration procedure.
	as_id	Application Server ID for which De-Registration result is received
	rtctx	Routing Context of the Application Server
	status	De-Registration status of the Application Server. The values for this parameter are same as “De-Registration Status” parameter defined in RFC 3332.
Operations		
Description	This API is used to collect management notifications from M3UA. This API must be invoked every time after an external event has been passed to the M3UA library. For example, this API must be invoked after M3UA_RECVMSG and M3UA_CHECK_TIMER API. A timer tick can result in an internal timer expiry that can be a notification to the layer management. This API must be repeatedly invoked until it returns failure. This API is used to notify state change of ASP/AS/Remote ASP/Remote AS/Connection/RK. It is also used to notify that an M3UA ERROR message or M3UA NTFY message has been received.	
Restrictions/ Bugs		
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/confasp1.c & comm/src/confsgp1.c, function name um3_process_mgmt_ntfy	

13 User APIs (USER→M3UA)

Following APIs are used by M3UA Users [Inter Working Function (IWF) or Nodal Interworking Function (NIF) at SG and ISUP/SCCP at AS].

13.1 M3UA_TRANSFER

Prototype	m3_s32 m3ua_transfer(m3_u16 user, m3ua_txr_t *pparam);	
Parameters	Associated C Data Structures	

```
typedef struct {
    m3_u32                  nw_app;
    m3_bool_t               add_rtctx;
    m3_u32                  crn_id;
    m3_rt_lbl_t             rt_lbl;
    m3_u32                  prot_data_len;
    m3_u8                   *p_prot_data;
} m3ua_txr_t;

typedef struct
{
    m3_u32                  opc;
    m3_u32                  dpc;
    m3_u8                   si;
    m3_u8                   ni;
    m3_u8                   mp;
    m3_u8                   sls;
} m3_rt_lbl_t;
```

	user	Identifier of user that is requesting transfer of SS7 message.
	nw_app	Network Appearance associated with the SS7 message.
	add_rtctx	This parameter indicates if routing context should be added to the DATA message. If this parameter is set to M3_FALSE then routing context is not added to the M3UA DATA message.
	crn_id	Correlation identifier associated with the DATA message. If this parameter is set to M3_MAX_U32 then correlation identifier is not added to the M3UA DATA message.
	rt_lbl	Routing Label of the SS7 message. It consists of OPC, DPC, Service Indicator, Network Indicator, message priority and Signaling Link Selection.
	prot_data_len	Length of SS7 user part message in bytes.
	p_prot_data	Pointer to the actual protocol message generated from the SS7 user part.
Operations		
Description	This API is used by SS7 user part to transfer a message to the appropriate destination. This primitive is equivalent to MTP-3 TRANSFER.	
Restrictions/ Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP_STATE	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_txr	

13.2 M3UA_PAUSE

Prototype	m3_s32 m3ua_pause(m3_u16 user, m3ua_pause_t *pparam) ;	

Parameters	Associated C Data Structures	typedef struct {
		    m3_u16 **num_pc**;
		    m3_u32 **pc_list**[M3_MAX_PC_SSNM];
		    m3_u32 **nw_app**;
		    m3_u8 **info_len**;
		    m3_u8 *__p_info_str__;
		    m3_u16 **num_rc**;
		    m3_u32 **rc_list**[M3_MAX_RTCTX];}
		m3ua_pause_t;

	user	Nodal Inter-working Function (NIF) at SGP indicating that it has received a Pause indication from the MTP-3.
	nw_app	Network Appearance associated with the Pause indication received at MTP-3.
	num_pc	Number of valid point codes present in the “pc_list” parameter.
	pc_list[]	List of point codes for which pause indication has been received by the MTP-3 at SGP.
	info_len	Length of the optional information string to be sent in the M3UA SSNM message. This parameter can be set to 0 if no information string is to be attached with the SSNM message. Maximum length of Information string can be 255 characters.
	p_info_str	Pointer to the information string to be sent in the SSNM message.
	num_rc	Number of routing contexts present in the routing context list (rc_list).
	rc_list	List of routing contexts for which traffic needs to be paused (blocked) through this Signaling Gateway.
Operations		
Description	This API is used by Nodal Inter-working Function (NIF) to indicate to the M3UA that MTP-3 at SGP has received a pause indication.	
Restrictions/ Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP_STATE	
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_pause	

13.3 M3UA_RESUME

Prototype	m3_s32 m3ua_resume(m3_u16 user, m3ua_resume_t *pparam);	

Parameters	Associated C Data Structures	typedef struct {
		    m3_u16 **num_pc**;
		    m3_u32 **pc_list**[M3_MAX_PC_SSNM];

		m3_u32 **nw_app**; m3_u8 **info_len**; m3_u8 ***p_info_str**; } m3ua_resume_t;
	user	Nodal Inter-working Function (NIF) at SGP indicating that it has received a Resume indication from the MTP-3.
	nw_app	Network Appearance associated with the Resume indication received at MTP-3.
	num_pc	Number of valid point codes present in the “pc_list” parameter.
	pc_list[]	List of point codes for which resume indication has been received by the MTP-3 at SGP.
	info_len	Length of the optional information string to be sent in the M3UA SSNM message. This parameter can be set to 0 if no information string is to be attached with the SSNM message. Maximum length of Information string can be 255 characters.
	p_info_str	Pointer to the information string to be sent in the SSNM message.
	num_rc	Number of routing contexts present in the routing context list (rc_list).
	rc_list	List of routing contexts for which traffic may be resumed through this Signaling Gateway.
Operations		
Description	This API is used by Nodal Inter-working Function (NIF) to indicate to the M3UA that MTP-3 at SGP has received a resume indication.	
Restrictions/ Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP_STATE	
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_resume	

13.4 M3UA_STATUS

Prototype	m3_s32 m3ua_status(m3_u16 user_id, m3ua_status_t *pparam);	
Parameters	Associated C Data Structures	typedef struct { m3_u8 **cause**; union { struct { m3_u32 **nw_app**; m3_u16 **num_pc**; m3_u32 **pc_list**[M3_MAX_PC_SSNM]; m3_u32 **crnd_dpc**; m3_u8 **cong_level**; m3_u8 **info_len**; m3_u8 ***p_info_str**; m3_u16 **num_rc**; m3_u32 **rc_list**[M3_MAX_RTCTX];

```
        } cong_inf;
        struct {
            m3_u32       nw_app;
            m3_u32       ptcode;
            m3_u8        user;
            m3_u8        info_len;
            m3_u8        *p_info_str;
            m3_u16       num_rc;
            m3_u32       rc_list[M3_MAX_RTCTX];
        } upu_inf;
        struct {
            m3_u32       nw_app;
            m3_u16       num_pc;
            m3_u32
pc_list[M3_MAX_PC_SSNM];
            m3_u8        info_len;
            m3_u8        *p_info_str;
            m3_u16       num_rc;
            m3_u32       rc_list[M3_MAX_RTCTX];
        } drst_inf;
    } status_inf;
} m3ua_status_t;
```

	user_id	Nodal Inter-working Function (NIF) at SGP indicating that it has received a Status indication from the MTP-3.
	cause	Cause associated with the status indication received by the MTP-3 at SGP. It can indicate User Part Unavailable, SS7 network congestion or Destination Restricted. The values for cause can be: • M3_CAUSE_UNKNOWN (0x00) • M3_CAUSE_UNEQUPPD_RMT_USR (0x01) • M3_CAUSE_INACCESS_RMT_USR (0x02) • M3_CAUSE_CONG (0x03) • M3_CAUSE_DRST (0x04) When cause is M3_CAUSE_CONG, information in “`cong_inf`” needs to be filled. When cause code is M3_CAUSE_DRST, information in “`drst_inf`” needs to be filled. For other cause codes, information in “`upu_inf`” needs to be filled.
	nw_app	Network Appearance associated with the Status indication received at MTP-3.
	num_pc	Number of valid point codes present in the “pc_list” parameter.
	pc_list[]	List of point codes for which status indication has been received by the MTP-3 at SGP.
	crnd_dpc	Concerned destination point code parameter present in the M3UA SCON message. This must be set to M3_MAX_U32 if this parameter is not required in the M3UA SCON message. This parameter is only set when M3UA SCON message is being sent from ASP to SGP.
	cong_level	Congestion level parameter in the M3UA SCON message. The

		congestion levels are as specified in RFC 3332.
	ptcode	Point code parameter in the M3UA DUPU message.
	user	SS7 user part identifier parameter in the M3UA DUPU message. The value of this field is encoded as per RFC 3332.
	info_len	Length of the optional information string to be sent in the M3UA SSNM message. This parameter can be set to 0 if no information string is to be attached with the SSNM message. Maximum length of Information string can be 255 characters.
	p_info_str	Pointer to the information string to be sent in the SSNM message.
	num_rc	Number of routing contexts present in the routing context list (rc_list).
	rc_list	List of routing contexts for which this particular status indication applies.
Operations		
Description	This API is used by Nodal Inter-working Function (NIF) to indicate to the M3UA that MTP-3 at SGP has received a resume indication.	
Restrictions/ Bugs	Presently this API does not allow sending SCON message from ASP to SGP, so “crnd_dpc” parameter must always be set to M3_MAX_U32.	
Pre-requisites	M3UA_INIT, M3UA_ASP_STATE	
Code Sample	Please refer source file comm/src/confsgp1.c, function name um3_m3ua_status	

13.5 M3UA_AUDIT

Prototype	m3_s32 m3ua_audit(m3_u16 user, m3ua_audit_t *pparam);	
Parameters	Associated C Data Structures	`typedef struct {` `m3_u16 num_pc;` `m3_u32 pc_list[M3_MAX_PC_SSNM];` `m3_u32 nw_app;` `m3_u8 info_len;` `m3_u8 *p_info_str;` `} m3ua_audit_t;`
	user	SS7 user part identifier that is requesting status of SS7 point codes.
	nw_app	Network appearance associated with the point codes whose reach-ability status is requested by the SS7 user part.
	num_pc	Number of valid point codes present in the “pc_list” parameter.
	pc_list[]	List of point codes whose reach-ability status is requested by the SS7 user part.
	info_len	Length of the optional information string to be sent in the M3UA SSNM message. This parameter can be set to 0 if no information string is to be attached with the SSNM message.
	p_info_str	Pointer to the information string to be sent in the SSNM message.
	num_rc	Number of routing contexts present in the routing context list

		(rc_list).
	rc_list	List of routing contexts for which status of point codes has been requested. Typically, when an AS becomes ACTIVE, this primitive may be invoked with the routing context of the AS that has become ACTIVE.
Operations		
Description	This API is used by SS7 user part to request status of SS7 point codes. This API sends M3UA DAUD message to all the relevant SGP. In response to DAUD message, SGP sends DUNA/DAVA/DUPU/SCON/DRST message(s) to the ASP.	
Restrictions/ Bugs	None	
Pre-requisites	M3UA_INIT, M3UA_ASP_STATE	
Code Sample	Please refer source file comm/src/confasp1.c, function name um3_m3ua_audit	

14 M3UA to SCTP API

14.1 M3UA_SENDMSG

Prototype	m3_s32 m3ua_sendmsg(m3_u32 assoc, m3_u32 stream, m3_u32 msglen, m3_u8 *pmsg);	
Parameters	assoc	SCTP Association identifier on which M3UA requires to transmit message. It is to be noted that this SCTP Association Identifier has already been provisioned in M3UA when a connection was added using M3UA_CONN API.
	stream	Stream Identifier of the outgoing stream on which message is to be transmitted.
	msglen	Length of message to be transmitted.
	pmsg	Pointer to the starting memory location where message is stored.
Operations		
Description	This API is used to transmit data on associations provided by the transport layer (SCTP) below M3UA. This API needs to be written and integrated with M3UA. It is responsibility of application developer to write this API using services provided by the underlying transport layer.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/sctp.c, function name m3ua_sendmsg	

15 SCTP to M3UA API

15.1 M3UA_RECVMSG

Prototype	m3_s32 m3ua_recvmsg(m3_u32 assoc, m3_u32 stream, m3_u32

	msglen, m3_u8 *pmsg);	
Parameters	assoc	SCTP Association identifier on which message has been received. It is to be noted that this SCTP Association Identifier has already been provisioned in M3UA when a connection was added using M3UA_CONN API.
	stream	Stream Identifier of the incoming stream on which message has been received.
	msglen	Length of received message.
	pmsg	Pointer to the starting memory location where message is stored.
Operations		
Description	This API is used by M3UA to receive data from transport layer below M3UA. This API should be called with appropriate parameters to pass a message from transport layer to M3UA.	
Restrictions/Bugs	None	
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/sctp.c, function name get_message	

16 User API (M3UA→USER)

16.1 M3UA_USER_NTFY

Prototype	m3_s32 m3ua_user_ntfy(m3ua_user_ntfy_t *);	

Parameters	Associated C Data Structures	`typedef struct {`
		`    m3_u8          type;`
		`    union {`
		`        struct {`
		`            m3_u16       user_id;`
		`            m3_u32       nw_app;`
		`            m3_u32       ptcode;`
		`        } pause;`
		`        struct {`
		`            m3_u16       user_id;`
		`            m3_u32       nw_app;`
		`            m3_u32       ptcode;`
		`        } resume;`
		`        struct {`
		`            m3_u16       user_id;`
		`            m3_u32       nw_app;`
		`            m3_u32       ptcode;`
		`            m3_u32       crnd_dpc;`
		`            m3_u8        cause;`
		`            m3_u8        inf;`
		`        } status;`
		`        struct {`
		`            m3_u16       user_id;`
		`            struct {`
		`                m3_u32            nw_app;`

```
                m3_u32                  rtctx;
                m3_u32                  crn_id;
                m3_rt_lbl_t             rt_lbl;
                m3_u16
prot_data_len;
                m3_u8
*p_prot_data;
            } inf;
        } transfer;
        struct {
            m3_u16        user_id;
            m3_u32        nw_app;
            m3_u8         mask;
            m3_u32        ptcode;
        } audit;
        struct {
            m3_u16        user_id;
            m3_u32        nw_app;
            m3_u32        ptcode;
            m3_rtstate_t  state;
            m3_u32        as_id;
        } rtstate;
    } param;
} m3ua_user_ntfy_t;
```

type	Type of management notification received from M3UA. This parameter is used to determine the type of indication and the set of valid parameters in the union. This is an unsigned 8 bit value and can take following values: • M3_USER_NTFY_PAUSE – PAUSE notification • M3_USER_NTFY_RESUME – RESUMT notification • M3_USER_NTFY_STATUS – STATUS notification • M3_USER_NTFY_TRANSFER – TRANSFER notification • M3_USER_NTFY_AUDIT – AUDIT notification
PAUSE Notification/ RESUME Notification	
user_id	User identifier for which notification has been sent by M3UA.
nw_app	Network Appearance associated with the notification.
ptcode	Point code associated with the notification.
STATUS Notification	
user_id	User identifier for which notification has been sent by M3UA.
nw_app	Network Appearance associated with the notification.
ptcode	Point code associated with the notification.
crnd_dpc	Concerned DPC parameter received in the M3UA SCON message.
cause	Cause associated with the STATUS indication.
Inf	Information present in the STATUS indication.
TRANSFER Notification	
user_id	User identifier for which notification has been sent by M3UA.

	nw_app	Network Appearance associated with the notification.
	rtctx	Routing context received in the Data message. It is useful for obtaining the local Application Server by which this message should be processed. Every Application Server has a user associated with it and routing context can be used to find associated user.
	crn_id	Correlation identifier received in the Data message.
	rt_lbl	Routing label received in the Data message.
	prot_data_len	Length of protocol data received in the Data message.
	p_prot_data	Pointer to protocol data received in the Data message.
	AUDIT Notification	
	user_id	User identifier for which notification has been sent by M3UA.
	nw_app	Network Appearance associated with the notification.
	ptcode	Point code associated with the notification.
	mask	This parameter is specifies the number of least significant bits of the point code to be masked to obtain a list of point codes whose status is requested by M3UA peer entity (ASP). For example; if mask is 0x04, and point code is 0x0121, then it means that status (reachable / unreachable) of all point codes from 0x0120 to 0x012F need to be reported back to M3UA peer entity.
	ROUTE-STATE Notification	
	user_id	User identifier for which notification has been sent by M3UA.
	nw_app	Network Appearance associated with the notification.
	ptcode	Point code associated with the notification.
	state	Route-State of the point code as applicable to the particular local AS. Following are the route states: • M3_RTSTATE_UP (0) • M3_RTSTATE_DOWN (1) • M3_RTSTATE_RESTRICTED (2) • M3_RTSTATE_CONG_1 (3) • M3_RTSTATE_CONG_2 (4) • M3_RTSTATE_CONG_3 (5)
	as_id	Specifies the AS-ID for which route state has changed
Operations		
Description	This API is used to collect user notifications from M3UA. This API must be invoked every time a message is received or timer tick is notified to M3UA. A timer tick can result in an internal timer expiry that can be a notification to the user. This API must be repeatedly invoked until it returns failure. This API is used to notify PAUSE/ RESUME/ STATUS/ TRANSFER to the MTP-3 User at ASP. This API is also used to notify AUDIT/STATUS/TRANSFER to the Nodal Inter-working Function (NIF) at SGP. AUDIT notification is used by M3UA to request the status of point codes from NIF.	
Restrictions/ Bugs		
Pre-requisites	M3UA_INIT	
Code	Please refer source file comm/src/confasp1.c & comm/src/confsgp1.c, function	

Sample	name um3_process_user_ntfy

17 Timing Management API

17.1 M3UA_TIMER_CHECK

Prototype	void * m3timer_ckexpiry(void *);	
Parameters		
Operations		
Description	This API must be called exactly M3_TICKS_PER_SEC times every second. This API calculates internal timer expiry based on the number of times it has been invoked. By default value of M3_TICKS_PER_SEC is set to 1 as granularity of M3UA timers is generally in seconds.	
Restrictions/Bugs		
Pre-requisites	M3UA_INIT	
Code Sample	Please refer source file comm/src/sctp.c, function name waitfor_message	

18 Data Structures Associated with the M3UA APIs

All the data structures & values that would be used by application are present in following M3UA include files:

1. **m3ua_defines.h**
2. **m3ua_types.h**
3. **m3ua_api.h**
4. **m3ua_errno.h**

Following is explanation for some of the key data structures:

(m3ua_defines.h, please see file in ./m3ua/inc for complete details)

```
/* Maximum value for standard data structure types */
#define M3_MAX_U32                              0xFFFFFFFF
#define M3_MAX_U16                              0xFFFF
#define M3_MAX_U8                               0xFF

#define M3_TRUE                                 1
#define M3_FALSE                                0

#define M3_NULL                                 0

/**************************** Managed Object operations ***************/

#define M3_ADD                                  1
#define M3_DELETE                               2
#define M3_GET                                  3
#define M3_MODIFY                               4

#define M3_REGISTER                             1
#define M3_DEREGISTER                           2
#define M3_STATUS                               3


/******** Transfer pool management ************************************/

#define M3_NUM_TXR_POOL_BUFF                    16
#define M3_MAX_TXR_BUFF_SIZE                    1024

#define M3_NUM_MGMT_POOL_BUFF                   16
#define M3_MAX_MGMT_BUFF_SIZE                   512

#define M3_NUM_PD_POOL_BUFF                     2048
#define M3_MAX_PD_BUFF_SIZE                     1024

/**************************** Message Class ****************************/
#define M3_MSG_CLASS_MGMT                       0x00
#define M3_MSG_CLASS_TXR                        0x01
#define M3_MSG_CLASS_SSNM                       0x02
#define M3_MSG_CLASS_ASPSM                      0x03
#define M3_MSG_CLASS_ASPTM                      0x04
#define M3_MSG_CLASS_RKM                        0x09
```

```
/************** Default Timer durations ****************/
#ifndef M3_TICKS_PER_SEC
#define M3_TICKS_PER_SEC                     1
#endif

#define M3_PD_TIMER_INT                        m3_timer_table.pdtimer.dur
#define M3_ASPM_TIMER_INT_LOW                  m3_timer_table.aspmtimer.l_dur
#define M3_ASPM_TIMER_INT_HIGH                 m3_timer_table.aspmtimer.h_dur
#define M3_HBEAT_TIMER_INT                     m3_timer_table.hbeattimer.dur
#define M3_RKM_TIMER_INT                       m3_timer_table.rkmtimer.dur

/************** Default Retry counts *****************/
#define M3_ASPM_RETRY_LOW                      m3_timer_table.aspmtimer.sw_try
#define M3_ASPM_MAX_RETRY                      m3_timer_table.aspmtimer.retry
#define M3_HBEAT_MAX_RETRY                     m3_timer_table.hbeattimer.retry
#define M3_RKM_MAX_RETRY                       m3_timer_table.rkmtimer.retry


/************** ASP states ***************/

#define M3_ASP_DOWN                          0
#define M3_ASP_INACTIVE                      1
#define M3_ASP_ACTIVE                        2

#define M3_ASP_DNSENT                        3
#define M3_ASP_UPSENT                        4

#define M3_ASP_ACSENT                        5
#define M3_ASP_IASENT                        6

#define M3_MAX_ASP_STATE                     5

/*************** Remote ASP states ***************/

#define M3_ASP_DOWN                             0
#define M3_ASP_INACTIVE                         1
#define M3_ASP_ACTIVE                           2

#define M3_ASP_DNRECV                           3
#define M3_ASP_UPRECV                           4

#define M3_MAX_ASP_STATE                     5

/*************** AS states ****************/

#define M3_AS_DOWN                           1
#define M3_AS_INACTIVE                       2
#define M3_AS_ACTIVE                         3
#define M3_AS_PENDING                        4

#define M3_MAX_AS_STATE                              4

/************** RK states ****************/

#define M3_RK_STATIC                         0
#define M3_RK_REG_IN_PROG                    1
#define M3_RK_REGD                           2
#define M3_RK_DEREG_IN_PROG                  3

/** Internal Registration/Deregistration status ****************/

#define M3_REG_STATUS_TIMEOUT                51
#define M3_DEREG_STATUS_TIMEOUT              101

/************** Types of management notifications **************/
#define M3_MGMT_NTFY_ASP_STATE                 1
#define M3_MGMT_NTFY_AS_STATE                  2
#define M3_MGMT_NTFY_R_ASP_STATE               3
#define M3_MGMT_NTFY_R_AS_STATE                4
#define M3_MGMT_NTFY_CONN_STATE                5
```

```
#define M3_MGMT_NTFY_NOTIFY                    6
#define M3_MGMT_NTFY_ERR                       7
#define M3_MGMT_NTFY_REGISTER                  8
#define M3_MGMT_NTFY_DEREGISTER                9
#define M3_MGMT_NTFY_REG_STATUS                10
#define M3_MGMT_NTFY_DEREG_STATUS              11

/************** Types of user notifications ****************/
#define M3_USER_NTFY_PAUSE                     1
#define M3_USER_NTFY_RESUME                    2
#define M3_USER_NTFY_STATUS                    3
#define M3_USER_NTFY_TRANSFER                  4
#define M3_USER_NTFY_AUDIT                     5

/*************** few imp. constants ****************/
#define M3_MAX_LOCAL_ROUTE                   256
#define M3_MAX_ROUTE                         256
#define M3_MAX_INFO_STR_LEN                  256
#define M3_MSG_HDR_LEN                       8

/************** User configurable defines ***************/

#ifndef M3_MAX_RTCTX
#define M3_MAX_RTCTX                         16
#endif

#ifndef M3_MAX_DPC_PER_RK
#define M3_MAX_DPC_PER_RK                    1
#endif

#ifndef M3_MAX_SI_PER_RK
#define M3_MAX_SI_PER_RK                     4
#endif

#ifndef M3_MAX_OPC_PER_RK
#define M3_MAX_OPC_PER_RK                    64
#endif

#ifndef M3_MAX_CKT_RANGE_PER_RK
#define M3_MAX_CKT_RANGE_PER_RK              16
#endif

#ifndef M3_MAX_TLV_VAL_SIZE
#define M3_MAX_TLV_VAL_SIZE                  256
#endif

#ifndef M3_MAX_PROT_DATA_LEN
#define M3_MAX_PROT_DATA_LEN                 512
#endif

#ifndef M3_MAX_PC_SSNM
#define M3_MAX_PC_SSNM                       64
#endif

#ifndef M3_MAX_RK_PER_RKM
#define M3_MAX_RK_PER_RKM                    16
#endif

#ifndef M3_MAX_ASPM_SESS_PER_CONN
#define M3_MAX_ASPM_SESS_PER_CONN            8
#endif

#ifndef M3_MAX_RKM_SESS_PER_CONN
#define M3_MAX_RKM_SESS_PER_CONN             4
#endif

#ifndef M3_MAX_RK
#define M3_MAX_RK                            M3_MAX_RK_PER_RKM
#endif
```

```
#ifndef M3_MAX_AS
#define M3_MAX_AS                             8
#endif

#ifndef M3_MAX_ASP
#define M3_MAX_ASP                            64
#endif

#ifndef M3_MAX_R_ASP
#define M3_MAX_R_ASP                          128
#endif

#ifndef M3_MAX_R_SGP
#define M3_MAX_R_SGP                          32
#endif

#ifndef M3_MAX_SGP
#define M3_MAX_SGP                            4
#endif

#ifndef M3_MAX_R_AS
#define M3_MAX_R_AS                           64
#endif

#ifndef M3_MAX_SG
#define M3_MAX_SG                             8
#endif

#ifndef M3_MAX_CONN
#define M3_MAX_CONN                           128
#endif

#ifndef M3_MAX_ASSOCID
#define M3_MAX_ASSOCID                        512
#endif

#ifndef M3_MAX_NWAPP
#define M3_MAX_NWAPP                          8
#endif

#ifndef M3_MAX_USR
#define M3_MAX_USR                            32
#endif

#ifndef M3_MAX_SGP_PER_SG
#define M3_MAX_SGP_PER_SG                     4
#endif

#ifndef M3_MAX_ROUTES_PER_SG
#define M3_MAX_ROUTES_PER_SG                  64
#endif

#ifndef M3_MAX_MGMT_NTFY
#define M3_MAX_MGMT_NTFY                      8
#endif

#ifndef M3_MAX_USER_NTFY
#define M3_MAX_USER_NTFY                      32
#endif

#ifndef M3_MAX_USER_DATA_SIZE
#define M3_MAX_USER_DATA_SIZE                 272
#endif

#ifndef M3_MAX_HBEAT_DATA_SIZE
#define M3_MAX_HBEAT_DATA_SIZE                256
#endif

#ifndef M3_MAX_TIMERS
#define M3_MAX_TIMERS                         256
```

```
#endif

#ifndef M3_DEF_R_AS_TRFMODE
#define M3_DEF_R_AS_TRFMODE                  M3_TRFMODE_BROADCAST
#endif

/************** Number of memory buffers available to M3UA **********/
/************** Scale these as per the application requirement **********/

#ifndef M3_NUM_32BYTE_BUFS
#define M3_NUM_32BYTE_BUFS    64
#endif

#ifndef M3_NUM_64BYTE_BUFS
#define M3_NUM_64BYTE_BUFS    256
#endif

#ifndef M3_NUM_128BYTE_BUFS
#define M3_NUM_128BYTE_BUFS   256
#endif

#ifndef M3_NUM_256BYTE_BUFS
#define M3_NUM_256BYTE_BUFS   256
#endif

#ifndef M3_NUM_512BYTE_BUFS
#define M3_NUM_512BYTE_BUFS   128
#endif

#ifndef M3_NUM_1024BYTE_BUFS
#define M3_NUM_1024BYTE_BUFS  64
#endif

#ifndef M3_NUM_2048BYTE_BUFS
#define M3_NUM_2048BYTE_BUFS  32
#endif

#ifndef M3_NUM_4096BYTE_BUFS
#define M3_NUM_4096BYTE_BUFS  16
#endif

#ifndef M3_NUM_8192BYTE_BUFS
#define M3_NUM_8192BYTE_BUFS  16
#endif
```

(m3ua_types.h, please see file in ./m3ua/inc for complete details)

```
/************* Basic Data types ***************/
typedef int                       m3_s32;
typedef unsigned int              m3_u32;
typedef short int                 m3_s16;
typedef unsigned short int        m3_u16;
typedef char                      m3_s8;
typedef unsigned char             m3_u8;


/************** Derived data types ****************/
typedef m3_u8                     m3_bool_t;
typedef m3_u32                    m3_traffic_mode_t;
typedef m3_u16                    m3_msg_type_t;
typedef m3_u32                    m3_sg_mode_t;
typedef m3_u8                     m3_asp_state_t;
typedef m3_u8                     m3_as_state_t;
typedef m3_u8                     m3_rk_state_t;
```

```
/************** Enumerated data types ****************/

/*** Types of SS7 standards supported by M3UA ********/

typedef enum {
    M3_STD_ANSI,
    M3_STD_ITU
} m3_standard_t;
/*** States of an SCTP association or M3UA connection *****/

typedef enum
{
    M3_CONN_NOT_ESTB      = 0,
    M3_CONN_SETUP_IN_PROG = 1,
    M3_CONN_ESTB          = 2,
    M3_CONN_CONG_1        = 3,
    M3_CONN_CONG_2        = 4,
    M3_CONN_CONG_3        = 5,
    M3_CONN_ALIVE         = 6
} m3_conn_state_t;

typedef m3_u32 m3_conn_opt_t;

/** States of M3UA Restart / MTP3 Restart ***/

typedef enum
{
    M3_RESTART_IN_PROGRESS    = 0,
    M3_RESTART_DONE           = 1
} m3_restart_status_t;

/** States of an SS7 network ***/

typedef enum
{
    M3_NW_UP              = 0,
    M3_NW_DOWN            = 1
} m3_nw_status_t;

/*** States associated with a Point Code **/

typedef enum
{
    M3_PC_UP,
    M3_PC_DOWN,
    M3_PC_RESTRICTED,
    M3_PC_CONG_1,
    M3_PC_CONG_2,
    M3_PC_CONG_3
} m3_pc_state_t;

/*** Route States towards a Point Code or a group of Point Codes ***/

typedef enum
{
    M3_RTSTATE_UP,
    M3_RTSTATE_DOWN,
    M3_RTSTATE_RESTRICTED,
    M3_RTSTATE_CONG_1,
    M3_RTSTATE_CONG_2,
    M3_RTSTATE_CONG_3
} m3_rtstate_t;

/** Various ASP configuration names that may be modified or retrieved ***/

typedef enum
{
    M3_ASP_M3ASP_ID,
    M3_ASP_NWAPP,
    M3_ASP_ADD_AS,
    M3_ASP_DEL_AS,
    M3_ASP_ADD_R_ASP,
    M3_ASP_DEL_R_ASP
} m3_asp_confname_t;

/** Various AS configuration names that may be modified or retrieved ***/

typedef enum {
    M3_AS_RTCTX,
    M3_AS_RKEY,
    M3_AS_INFO
} m3_as_confname_t;

/** Various Remote AS configuration names that may be modified or retrieved **/

typedef enum {
    M3_R_AS_RTCTX,
    M3_R_AS_RKEY,
    M3_R_AS_MIN_ACT_ASP,
    M3_R_AS_INFO
}m3_r_as_confname_t;

/** Various SGP configuration names that may be modified or retrieved ***/

typedef enum {
    M3_SGP_NWAPP,
    M3_SGP_ADD_R_ASP,
    M3_SGP_DEL_R_ASP
} m3_sgp_confname_t;

/** Various SG configuration names that may be modified or retrieved ***/

typedef enum {
    M3_SG_MODE,
    M3_SG_SGP_LIST,
    M3_SG_INFO
} m3_sg_confname_t;

/** Various Connection configuration names that may be modified or retrieved ***/

typedef enum {
    M3_CONN_ASSOC,
    M3_CONN_I_STR,
    M3_CONN_O_STR
} m3_conn_confname_t;

/****************** structures *********************/

/*** Circuit Range for ISUP based Routing Labels **/
typedef struct
{
    m3_u32                opc;
    m3_u16                lcic;
    m3_u16                ucic;
} m3_ckt_range_t;

/** AS Routing Key elements ***/
typedef struct {
    m3_u32                dpc;
    m3_u16                num_si;
    m3_u8                 si_list[M3_MAX_SI_PER_RK];
    m3_u16                num_opc;
    m3_u32                opc_list[M3_MAX_OPC_PER_RK];
    m3_u16                num_ckt_range;
    m3_ckt_range_t        ckt_range[M3_MAX_CKT_RANGE_PER_RK];
} m3_rk_elements_t;

/**** Routing Key Information ***/
```

```
typedef struct
{
    m3_traffic_mode_t     trfmode;
    m3_u32                nw_app;
    m3_u8                 num_rtparam;
    m3_rk_elements_t      rtparam[M3_MAX_DPC_PER_RK];
} m3_rk_inf_t;

/**** Point Code information ****/
typedef struct
{
    m3_u32                ptcode;
    m3_u32                nw_app;
} m3_pc_inf_t;
/**** Routing Label Information ****/
typedef struct
{
    m3_u32               opc;
    m3_u32               dpc;
    m3_u8                si;
    m3_u8                ni;
    m3_u8                mp;
    m3_u8                sls;
} m3_rt_lbl_t;


/*** Various types of Logging supported by M3UA ****/
typedef enum {
    m3uaErrorTrace     = 0x01,
    m3uaConfigTrace    = 0x02,
    m3uaAspmTrace      = 0x04,
    m3uaRkmTrace       = 0x08,
    m3uaSsnmTrace      = 0x10,
    m3uaTxrTrace       = 0x20,
    m3uaMgmtTrace      = 0x40,
    m3uaInMsgTrace     = 0x80,
    m3uaOutMsgTrace    = 0x0100,
    m3uaStartupTrace   = 0x0200,
    m3uaTimerTrace      = 0x0400,
    m3uaMaxTrcType
} m3TrcType_t;
```

(m3ua_api.h, see file in ./m3ua/inc for complete details)

```
/*************** configuration structures ****************/

typedef struct
{
    m3_u32                m3asp_id;
    m3_u32                sctp_ep_id;
    m3_bool_t             as_list[M3_MAX_AS];
    m3_u32                def_nwapp;
    struct {
        m3_bool_t         asp_list[M3_MAX_R_ASP];
    } r_as_inf[M3_MAX_R_AS];
} m3_asp_conf_t;

typedef struct
{
    m3_u32                sctp_ep_id;
} m3_r_asp_conf_t;

typedef struct
{
    m3_u32                rtctx;
    m3_rk_inf_t           rkey;
} m3_as_conf_t;

typedef struct
```

```
{
    m3_u32                  rtctx;
    m3_rk_inf_t             rkey;
    m3_u8                   min_act_asp;
} m3_r_as_conf_t;

typedef struct
{
    m3_u32                  assoc_id;
    m3_u32                  i_str;
    m3_u32                  o_str;
} m3_conn_conf_t;

typedef struct
{
    m3_pc_inf_t             pc_inf;
    m3_u16                  user_id;
} m3_local_rt_conf_t;

typedef struct
{
    m3_u32                  sctp_ep_id;
    m3_u32                  def_nwapp;
    struct {
        m3_bool_t           asp_list[M3_MAX_R_ASP];
    } r_as_inf[M3_MAX_R_AS];
} m3_sgp_conf_t;

typedef struct
{
    m3_u32                  sctp_ep_id;
} m3_r_sgp_conf_t;

typedef struct
{
    m3_sg_mode_t            sgmode;
    m3_u8                   num_sgp;
    m3_u32                  sgp_list[M3_MAX_SGP_PER_SG];
} m3_sg_conf_t;

typedef struct
{
    m3_pc_inf_t             pc_inf;
    m3_u32                  le_id;
    m3_u8                   priority;
} m3_rt_conf_t;

typedef struct {
    m3_u32                  nw_app;
    m3_standard_t           standard;
} m3_nwapp_conf_t;

typedef struct
{
    m3_u32                  sp_id;
    union {
        struct {
            m3_u8           sio;
            m3_u32          as_id;
        } mtp_user;
        struct {
            m3_bool_t       a_data;
        } nif_user;
    } user;
} m3_usr_conf_t;

/************** Configuration/management API structures **************/

typedef union {
    struct {
        m3_asp_conf_t info;
    } add;
    struct {
        m3_asp_conf_t info;
    } get;
    struct {
        m3_asp_confname_t    confname;
        m3_asp_conf_t        info;
        m3_u32               as_id;
        m3_u32               ras_id;
        m3_u32               rasp_id;
    } modify;
} m3ua_asp_t;

typedef union {
    struct {
        m3_as_conf_t info;
    } add;
    struct {
        m3_as_conf_t info;
    } get;
    struct {
        m3_as_confname_t    confname;
        m3_as_conf_t        info;
    } modify;
} m3ua_as_t;

typedef union {
    struct {
        m3_r_asp_conf_t info;
    } add;
    struct {
        m3_r_asp_conf_t info;
    } get;
} m3ua_r_asp_t;

typedef union {
    struct {
        m3_r_as_conf_t info;
    } add;
    struct {
        m3_r_as_conf_t info;
    } get;
    struct {
        m3_r_as_confname_t    confname;
        m3_r_as_conf_t        info;
    } modify;
} m3ua_r_as_t;

typedef union {
    struct {
        m3_sgp_conf_t info;
    } add;
    struct {
        m3_sgp_conf_t info;
    } get;
    struct {
        m3_sgp_confname_t    confname;
        m3_sgp_conf_t        info;
        m3_u32               ras_id;
        m3_u32               rasp_id;
    } modify;
} m3ua_sgp_t;

typedef union {
    struct {
        m3_r_sgp_conf_t info;
    } add;
    struct {
        m3_r_sgp_conf_t info;
    } get;
} m3ua_r_sgp_t;

typedef union {
    struct {
        m3_sg_conf_t info;
    } add;
    struct {
        m3_sg_conf_t info;
    } get;
    struct {
        m3_sg_confname_t    confname;
        m3_sg_conf_t        info;
    } modify;
} m3ua_sg_t;

typedef union {
    struct {
        struct {
            m3_u32        as_id;
        } info;
        m3_asp_state_t    state;
    } get;
    struct {
        struct {
            m3_u16        num_as;
            m3_u32        as_list[M3_MAX_AS];
        }info;
        m3_asp_state_t    state;
    } modify;
} m3ua_asp_state_t;

typedef union {
    struct {
        struct {
            m3_u32            as_id;
        } info;
        m3_asp_state_t    state;
    } get;
} m3ua_r_asp_state_t;

typedef union {
    struct {
        m3_as_state_t    state;
    } get;
} m3ua_r_as_state_t;

typedef union {
    struct {
        m3_conn_conf_t info;
    } add;
    struct {
        m3_conn_conf_t info;
    } get;
    struct {
        m3_conn_confname_t    confname;
        m3_conn_conf_t        info;
    } modify;
} m3ua_conn_t;

typedef union {
    struct {
        m3_conn_state_t state;
    } get;
    struct {
        m3_conn_state_t state;
    } modify;
} m3ua_conn_state_t;

typedef union {
    struct {
        m3_conn_opt_t opt;
    } get;
    struct {
        m3_conn_opt_t opt;
    } modify;
} m3ua_conn_opt_t;

typedef union {
    struct {
        m3_rt_conf_t       info;
    } add;
    struct {
        m3_rt_conf_t       info;
    } del;
} m3ua_route_t;

typedef union {
    struct {
        m3_local_rt_conf_t   info;
    } add;
    struct {
        m3_local_rt_conf_t   info;
    } del;
} m3ua_local_route_t;

typedef union {
    struct {
        m3_usr_conf_t    info;
    } add;
} m3ua_user_t;

typedef union {
    struct {
        m3_nwapp_conf_t    info;
    } add;
    struct {
        m3_nwapp_conf_t    info;
    } del;
} m3ua_nwapp_t;

typedef struct {
    m3_timer_type_t        type;
    union {
        m3_aspmtimer_t     aspmtimer;
        m3_pdtimer_t       pdtimer;
        m3_hbeattimer_t    hbeattimer;
        m3_rkmtimer_t      rkmtimer;
    } get;
    union {
        m3_aspmtimer_t     aspmtimer;
        m3_pdtimer_t       pdtimer;
        m3_hbeattimer_t    hbeattimer;
        m3_rkmtimer_t      rkmtimer;
    } modify;
} m3ua_timer_t;

typedef union {
    struct {
        m3_u32             asp_id;
        m3_u32             rsp_id;
        m3_u16             num_as;
        m3_u32             as_list[M3_MAX_AS];
    } reg;
    struct {
        m3_u32             asp_id;
        m3_u32             rsp_id;
        m3_u16             num_as;
        m3_u32             as_list[M3_MAX_AS];
    } dereg;
    struct {
        m3_u32              lsp_id;
        m3_u32              rasp_id;
        m3_u16              num_result;
        struct {
            m3_u32          as_id;
            m3_u32          reg_status;
        } result[M3_MAX_RK];
    } status;
} m3ua_rkey_t;

/*************** notification from M3UA to management ****************/
typedef struct {
    m3_u8          type;
    union {
        struct {
            m3_u32            lsp_id;
            m3_u32            rsp_id;
            m3_u32            err_code;
            m3_u16            num_rc;
            m3_u32            rc_list[M3_MAX_RTCTX];
            m3_u16            num_pc;
            m3_u32            pc_list[M3_MAX_PC_SSNM];
            m3_u32            nw_app;
            m3_u16            diag_len;
            m3_u8             *p_diag_inf;
        } err;
        struct {
            m3_u32            asp_id;
            m3_u32            rsp_id;
            m3_u32            as_id;
            m3_asp_state_t    state;
        } asp;
        struct {
            m3_u32            asp_id;
            m3_u32            lsp_id;
            m3_u32            as_id;
            m3_asp_state_t    state;
        } r_asp;
        struct {
            m3_u32            as_id;
            m3_u32            lsp_id;
            m3_u32            rsp_id;
            m3_as_state_t     state;
        } as;
        struct {
            m3_u32            as_id;
            m3_u32            lsp_id;
            m3_as_state_t     state;
        } r_as;
        struct {
            m3_u32            lsp_id;
            m3_u32            rsp_id;
            m3_conn_state_t   state;
        } conn;
        struct {
            m3_u32            lsp_id;
            m3_u32            rsp_id;
            m3_u16            status_type;
            m3_u16            status_inf;
            m3_u32            m3asp_id;
            m3_u16            num_as;
            m3_u32            as_list[M3_MAX_RTCTX];
        } notify;
        struct {
            m3_u32            lsp_id;
            m3_u32            asp_id;
            m3_u16            num_as;
            m3_u32            as_list[M3_MAX_RK];
        } reg;
        struct {
            m3_u32           lsp_id;
            m3_u32           asp_id;
            m3_u16           num_as;
            m3_u32           as_list[M3_MAX_RK];
        } dereg;
        struct {
            m3_u32           asp_id;
            m3_u32           rsp_id;
            m3_u16           num_as;
            struct {
                m3_u32       as_id;
                m3_u32       rtctx;
                m3_u32       status;
            } result[M3_MAX_RK];
        } reg_status;
        struct {
            m3_u32           asp_id;
            m3_u32           rsp_id;
            m3_u16           num_as;
            struct {
                m3_u32       as_id;
                m3_u32       rtctx;
                m3_u32       status;
            } result[M3_MAX_RK];
        } dreg_status;
    } param;
} m3ua_mgmt_ntfy_t;

/*************** User API structures ******************/

typedef struct {
    m3_u32               nw_app;
    m3_bool_t            add_rtctx;
    m3_u32               crn_id;
    m3_rt_lbl_t          rt_lbl;
    m3_u32               prot_data_len;
    m3_u8                *p_prot_data;
} m3ua_txr_t;

typedef struct {
    m3_u16               num_pc;
    m3_u32               pc_list[M3_MAX_PC_SSNM];
    m3_u32               nw_app;
    m3_u8                info_len;
    m3_u8                *p_info_str;
} m3ua_pause_t;

typedef struct {
    m3_u16               num_pc;
    m3_u32               pc_list[M3_MAX_PC_SSNM];
    m3_u32               nw_app;
    m3_u8                info_len;
    m3_u8                *p_info_str;
} m3ua_audit_t;

typedef struct {
    m3_u16               num_pc;
    m3_u32               pc_list[M3_MAX_PC_SSNM];
    m3_u32               nw_app;
    m3_u8                info_len;
    m3_u8                *p_info_str;
} m3ua_resume_t;

typedef struct {
    m3_u8                cause;
    union {
        struct {
            m3_u32       nw_app;
            m3_u16       num_pc;
            m3_u32      pc_list[M3_MAX_PC_SSNM];
            m3_u32      crnd_dpc;
            m3_u8       cong_level;
            m3_u8       info_len;
            m3_u8       *p_info_str;
        } cong_inf;
        struct {
            m3_u32      nw_app;
            m3_u32      ptcode;
            m3_u8       user;
            m3_u8       info_len;
            m3_u8       *p_info_str;
        } upu_inf;
        struct {
            m3_u32      nw_app;
            m3_u16      num_pc;
            m3_u32      pc_list[M3_MAX_PC_SSNM];
            m3_u8       info_len;
            m3_u8       *p_info_str;
        } drst_inf;
    } status_inf;
} m3ua_status_t;

typedef struct {
    m3_u32                nw_app;
    m3_restart_status_t  status;
} m3ua_restart_t;


/*************** notification from M3UA to User ****************/
typedef struct {
    m3_u8         type;
    union {
        struct {
            m3_u16       user_id;
            m3_u32       nw_app;
            m3_u32       ptcode;
        } pause;
        struct {
            m3_u16       user_id;
            m3_u32       nw_app;
            m3_u32       ptcode;
        } resume;
        struct {
            m3_u16       user_id;
            m3_u32       nw_app;
            m3_u32       ptcode;
            m3_u32       crnd_dpc;
            m3_u8        cause;
            m3_u8        inf;
        } status;
        struct {
            m3_u16       user_id;
            struct {
                m3_u32              nw_app;
                m3_u32              rtctx;
                m3_u32              crn_id;
                m3_rt_lbl_t         rt_lbl;
                m3_u16              prot_data_len;
                m3_u8               *p_prot_data;
            } inf;
        } transfer;
        struct {
            m3_u16       user_id;
            m3_u32       nw_app;
            m3_u8        mask;
            m3_u32       ptcode;
        } audit;
    } param;
} m3ua_user_ntfy_t;
```

```
/**************** M3UA API Prototypes ******************/

m3_s32 m3ua_init(void);

m3_s32 m3ua_as(m3_u32, m3_u8, m3ua_as_t *);

m3_s32 m3ua_asp(m3_u32, m3_u8, m3ua_asp_t *);

m3_s32 m3ua_asp_state(m3_u32, m3_u32, m3_u8, m3ua_asp_state_t *);

m3_s32 m3ua_conn(m3_u32, m3_u32, m3_u8, m3ua_conn_t *);

m3_s32 m3ua_conn_state(m3_u32, m3_u32, m3_u8, m3ua_conn_state_t *);

m3_s32 m3ua_conn_opt(m3_u32, m3_u32, m3_u8, m3ua_conn_opt_t *);

m3_s32 m3ua_local_route(m3_u8, m3ua_local_route_t *);

m3_s32 m3ua_mgmt_ntfy(m3ua_mgmt_ntfy_t *);

m3_s32 m3ua_nwapp(m3_u8, m3ua_nwapp_t *);

m3_s32 m3ua_r_as(m3_u32, m3_u8, m3ua_r_as_t *);

m3_s32 m3ua_r_as_state(m3_u32, m3_u32, m3_u8, m3ua_r_as_state_t *);

m3_s32 m3ua_r_asp(m3_u32, m3_u8, m3ua_r_asp_t *);

m3_s32 m3ua_r_asp_state(m3_u32, m3_u32, m3_u8, m3ua_r_asp_state_t *);

m3_s32 m3ua_route(m3_u8, m3ua_route_t *);

m3_s32 m3ua_r_asplock(m3_u32, m3_u8, void *);

m3_s32 m3ua_timer(m3_u8, m3ua_timer_t *);

m3_s32 m3ua_heartbeat(m3_u32, m3_u32, m3_u8, void *);

m3_s32 m3ua_r_sgp(m3_u32, m3_u8, m3ua_r_sgp_t *);

m3_s32 m3ua_sg(m3_u32, m3_u8, m3ua_sg_t *);

m3_s32 m3ua_sgp(m3_u32, m3_u8, m3ua_sgp_t *);

m3_s32 m3ua_user(m3_u16, m3_u8, m3ua_user_t *);

m3_s32 m3ua_pause(m3_u16, m3ua_pause_t *);

m3_s32 m3ua_resume(m3_u16, m3ua_resume_t *);

m3_s32 m3ua_status(m3_u16, m3ua_status_t *);

m3_s32 m3ua_transfer(m3_u16, m3ua_txr_t *);

m3_s32 m3ua_audit(m3_u16, m3ua_audit_t *);

m3_s32 m3ua_user_ntfy(m3ua_user_ntfy_t *);

void* m3timer_ckexpiry(void *);

/************** Transport Layer API Prototypes ************/

m3_s32 m3ua_sendmsg(m3_u32, m3_u32, m3_u32, m3_u8 *);

m3_s32 m3ua_recvmsg(m3_u32, m3_u32, m3_u32, m3_u8 *);

/************** Traces API ********************************/
```

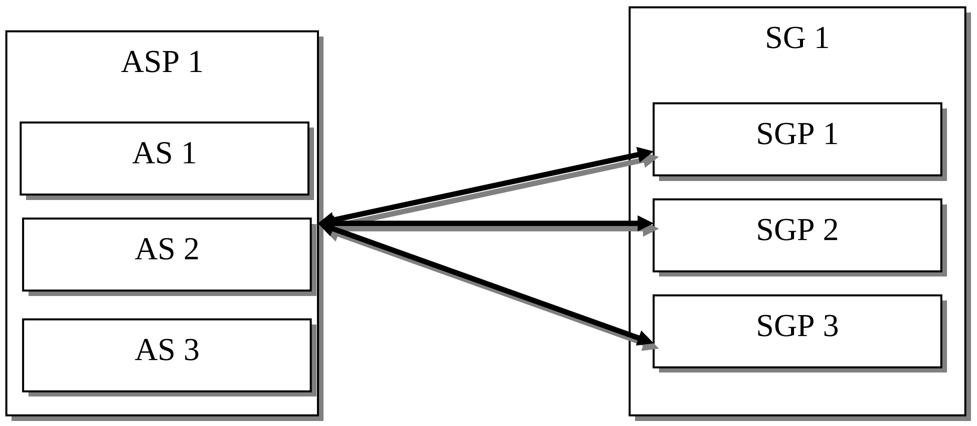
```
void m3ua_set_trace_map(m3_u32 aTraceMap);

void m3ua_add_trace(m3TrcType_t aTraceType);

void m3ua_del_trace(m3TrcType_t aTraceType);

m3_u32 m3ua_get_trace_map(void);
```

19 Supported Configurations

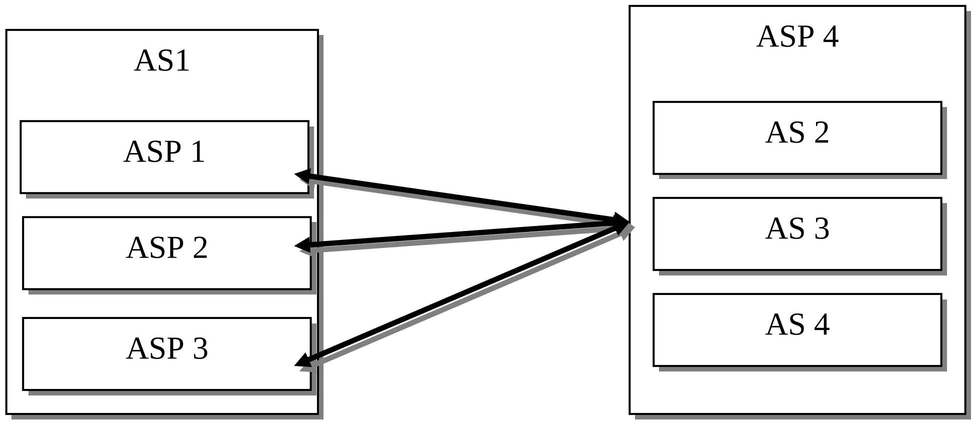
Some useful configurations that are supported by the M3UA are depicted in the figures shown in this section. Multiple ASP and multiple SGP can be configured in the same M3UA layer simultaneously. This makes possible to design a co-located ASP-SGP/ASP-ASP configuration using same instance of transport layer.

Some possible configurations supported are:

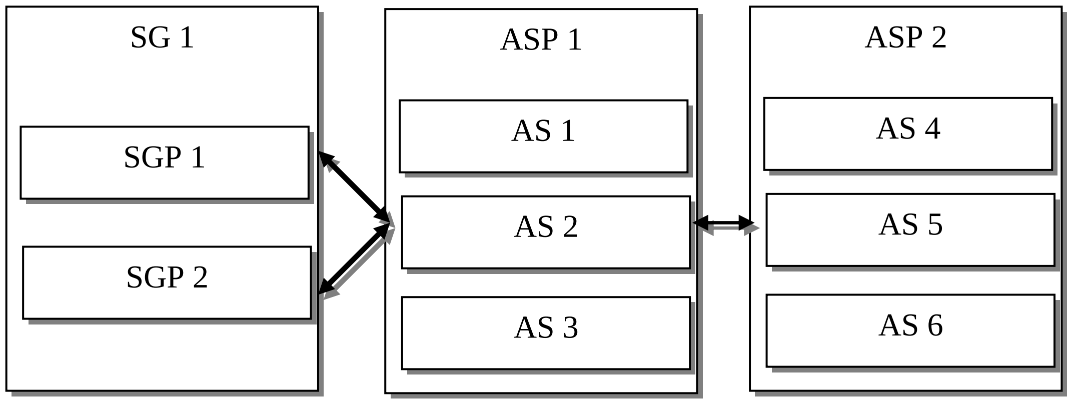
- Multiple SGP serving one SG
- Multiple ASP serving one AS
- One ASP serving multiple AS
- Load-sharing between multiple ASP serving same AS
- Load-sharing between multiple SGP serving same SG
- Primary Backup configuration for multiple ASP serving same AS
- Broadcast mode configuration for multiple SGP serving same SG



ASP 1
AS 1
AS 2
AS 3
SG 1
SGP 1
SGP 2
SGP 3



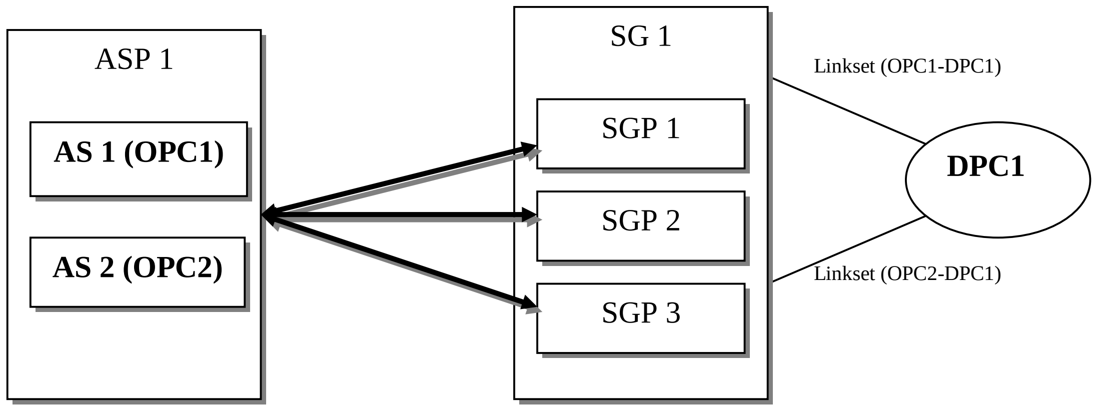
AS1
ASP 1
ASP 2
ASP 3
ASP 4
AS 2
AS 3
AS 4



SG 1
SGP 1
SGP 2
ASP 1
AS 1
AS 2
AS 3
ASP 2
AS 4
AS 5
AS 6

20 Description of SS7 Routes per Routing RC

As per RFC 4666, routing context list has become a conditional parameter in DUNA, DAVA, SCON, DRST and DUPU messages. This feature is useful only when an SCTP association is used for carrying traffic belonging to more than one routing context. In such case, it becomes necessary that Signaling Gateway is able to report SS7 route states on a per routing context basis. Following figure explains the situation better:



If Linkset (OPC1-DPC1) goes faulty or unavailable for some reason, then SG1 needs to send a PAUSE (DUNA message) to ASP1 saying that DPC1 is unavailable for OPC1. Here OPC1 is represented by the routing context of AS1.

Also, it is responsibility of the ASP1 to initiate audits at regular intervals for all the destination point codes to which it is sending traffic. In the above figure DAUD would be sent only for DPC1. The DAUD message would contain Routing context for both AS1 and AS2 as both AS1 and AS2 would be sending traffic to DPC1 through SG1.

To use this feature, following procedure is to be used:

1. Enable RT-PER-LRC feature by invoking M3UA_FEATURE API with feature bitmap set to 0x00000001 and operation set to M3_ADD.
2. While adding Signaling Gateway configuration, set “rt_per_rc_supp” to 1 for those SG that support reporting of SS7 routes on per routing context basis.
3. Signaling Gateway application would use parameters “num_rc” and “rc_list” when invoking M3UA_PAUSE, M3UA_RESUME and “M3UA_STATUS” API.

4. ASP side applications would use parameters “num_rc” and “rc_list” when invoking M3UA_AUDIT API
5. ASP side applications would receive SS7 route states on per “Application-Server (or RC)” basis in M3UA_USER_NTFY_RTSTATE.
6. M3UA_USER_NTFY_PAUSE, M3UA_USER_NTFY_RESUME, M3UA_USER_NTFY_STATUS and M3UA_USER_NTFY_AUDIT notifications contain “num_rc” and “rc_list” parameters.

21 Sample Application

This section explains the procedure of building and running sample application. It also explains working of the sample application. This application has been provided to help developers to write applications operating over M3UA.

The application first initializes transport layer, which is in UDP (or SCTP) in this case. M3UA has been written carefully such that any transport layer can be used very easily. The design philosophy applied here assumes that transport layer is a medium to send/receive messages to the peer only. This design burdens Layer Management / Wrapper Application / Thin Layer with the following additional tasks:

- Configuring transport layer [SCTP Layer]
- Creating M3UA related context in the transport layer if required
- Creating/Deleting transport level associations
- Handling notifications from transport layer & passing them to M3UA layer using appropriate primitives
- Passing requests from M3UA layer to the transport layer

But this provides a very high level of flexibility to M3UA to adapt any available transport layer very quickly and easily. The sample application has been written using UDP (and SCTP) as the transport layer.

Sample application contain functions with name um3_<API NAME>, where <API NAME> refers to a M3UA API. These functions demonstrate the usage of the APIs and the way in which code should be written to invoke a M3UA API. For example um3_m3ua_asp demonstrates the code for the usage of API M3UA_ASP.

A single threaded application has been implemented to avoid the overhead of mutual exclusion. Once SCTP associations (transport level connections) are created, then all the managed objects are configured in M3UA.

After configuring various objects in M3UA, ASPM procedure to make ASP INACTIVE is initiated. As soon as notification is received from M3UA suggesting

ASP state changed from DOWN to INACTIVE, ASPM procedure to make ASP ACTIVE is started.

After ASP moves to ACTIVE STATE, M3UA user APIs (TRANSFER, PAUSE, RESUME or STATUS) can be invoked to start M3UA user level procedures. Transfer of user data has been demonstrated in the sample application.

ASP state is changed from ACTIVE to DOWN after transferring user data. Procedures between ASP and SGP as well as ASP and ASP (IPSP-IPSP) are demonstrated in the sample application.

21.1 Building the Components of Sample Application

Sample application consists of 3 components: - ASP1, ASP2 and SGP1. The configuration of the sample application is depicted in config.txt file. To build these different components (when SCTP is not installed) following commands should be used: -

- make asp1
- make asp2
- make sgp1

or (if you have LK-SCTP 1.0.6 installed with your GNU OS / Linux)

- make sctp

This will build 3 executables asp1, asp2 and sgp1. To execute the application, run these components as per the order below using the specified commands.

- sgp1 sgp1=<SGP1 IP address>:<SGP1 UDP Port> asp1=<ASP1 IP address>:<ASP1 UDP Port>
- asp1 asp1=<ASP1 IP address>:<ASP1 UDP Port> asp2=<ASP2 IP address>:<ASP2 UDP Port> sgp1=<SGP1 IP address>:<SGP1 UDP Port>
- asp2 asp2=<ASP2 IP address>:<ASP2 UDP Port> asp1=<ASP1 IP address>:<ASP1 UDP Port>

Example: -

- sgp1 sgp1=127.0.0.1:4552 asp1=127.0.0.1:4550
- asp1 asp1=127.0.0.1:4550 asp2=127.0.0.1:4551 sgp1=127.0.0.1:4552
- asp2 asp2=127.0.0.1:4551 asp1=127.0.0.1:4550

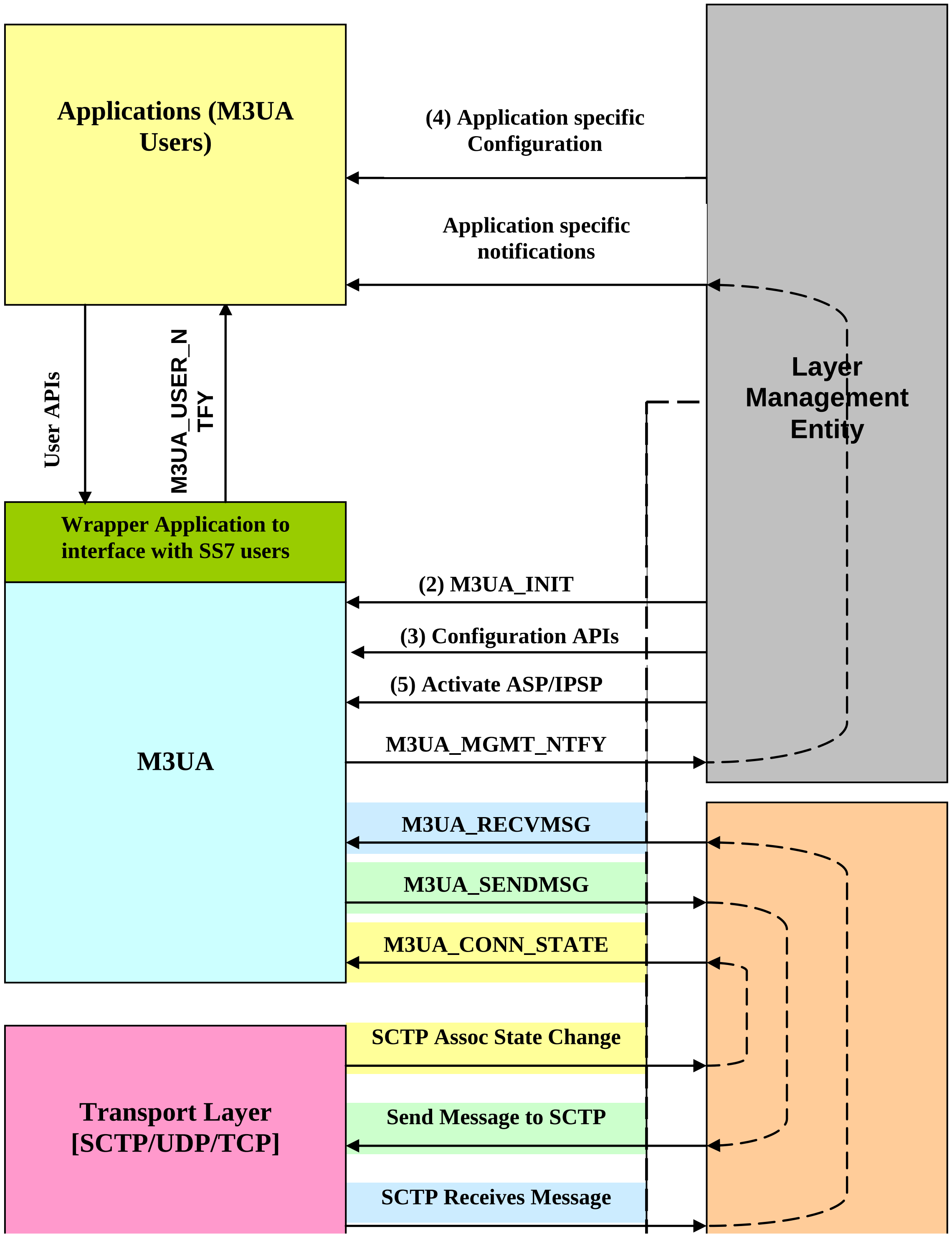
22 Integrating M3UA with Third Party Components (SCTP and SS7 User Parts)

A layer management entity would be required to first configure SCTP followed by M3UA followed by SS7 user parts. Layer management entity would also handle various notifications from M3UA layer and translate them into relevant information for SS7 user parts.

A wrapper application (light weight application) around M3UA layer shall be written to communicate with the SS7 users. This wrapper application would pass primitives between SS7 users and M3UA. It will also perform any conversion of primitives required to facilitate communication between M3UA and SS7 users. For example a MTP_TRANSFER primitive would be converted into M3UA_TRANSFER primitive by this wrapper application & M3UA_USER_NTFY (TRANSFER IND.) would be converted into MTP_TRANSFER_IND for SS7 users.

A thin layer between SCTP and M3UA would facilitate passing of messages between these SCTP and M3UA layers. Also, any SCTP association state change notifications need to be informed to the M3UA layer via this thin layer.

Following illustration shows how M3UA may be integrated with SCTP (transport layer) and SS7 user parts.



Applications (M3UA Users)
(4) Application specific Configuration
Application specific notifications
Layer Management Entity
User APIs
M3UA_USER_NTFY
Wrapper Application to interface with SS7 users
(2) M3UA_INIT
(3) Configuration APIs
(5) Activate ASP/IPSP
M3UA_MGMT_NTFY
M3UA
M3UA_RECVMSG
M3UA_SENDMSG
M3UA_CONN_STATE
SCTP Assoc State Change
Transport Layer [SCTP/UDP/TCP]
Send Message to SCTP
SCTP Receives Message

23 New features in Release 2.0.0

1. Memory management implemented
2. Timer management improved
3. Routing Key registration procedure implemented
4. Handling of ASP Failure notification
5. Basic trace management added.
6. And, lot of bugs fixed.

24 Release Notes – M3UA v 3.0.0

Bug fixes

1) Configuration modifications to SG
2) Correction of ASP1 sample application.
3) Correction of Route Status Update when multiple Point Code Status changes (m3uaSetRouteStatex)
4) Correction of SGP Index while sending DAUD using m3ua_audit
5) Correction of DAUD (AUDIT) handling on SGP.

New Additions / Features

1) Default Route State is DOWN when new Remote AS/SG is added. This makes it mandatory for ASP to initiate Auditing when it becomes ACTIVE in a certain SG.
2) Addition of Error Traces when Error Code is set
3) Route State update when ASP goes DOWN.

25 Release Notes – M3UA v 3.0.4

New Additions / Features

1) Inclusion of optional parameters Network appearance, routing context, ASP ID and traffic mode in M3UA messages may be controlled on per connection basis.