Aculab SS7

Maintenance API Guide

Revision 6.14.0

PROPRIETARY INFORMATION

Document Revision

Rev	Date	By	Detail
6.9.0	20.06.07	DSL	First issue
6.10.1	12.09.08	DSL	TCAP & M3UA support
6.10.3	30.10.08	DSL	Minor updates
6.10.4	15.12.08	DSL	Addition of MTP3 API functions.
6.11.0	06.09.10	DSL	Statistics and additional MTP3 requests.
6.11.11	01.11.11	DSL	Minor corrections and clarifications.
6.12.2	28.06.13	DSL	Remove Solaris, add IPv6.
6.13.0	27.10.14	DSL	Minor updates
6.14.0	15.09.16	DSL	Minor updates

CONTENTS

1 Introduction

This document describes the SS7 maintenance API and the MTP3 API.

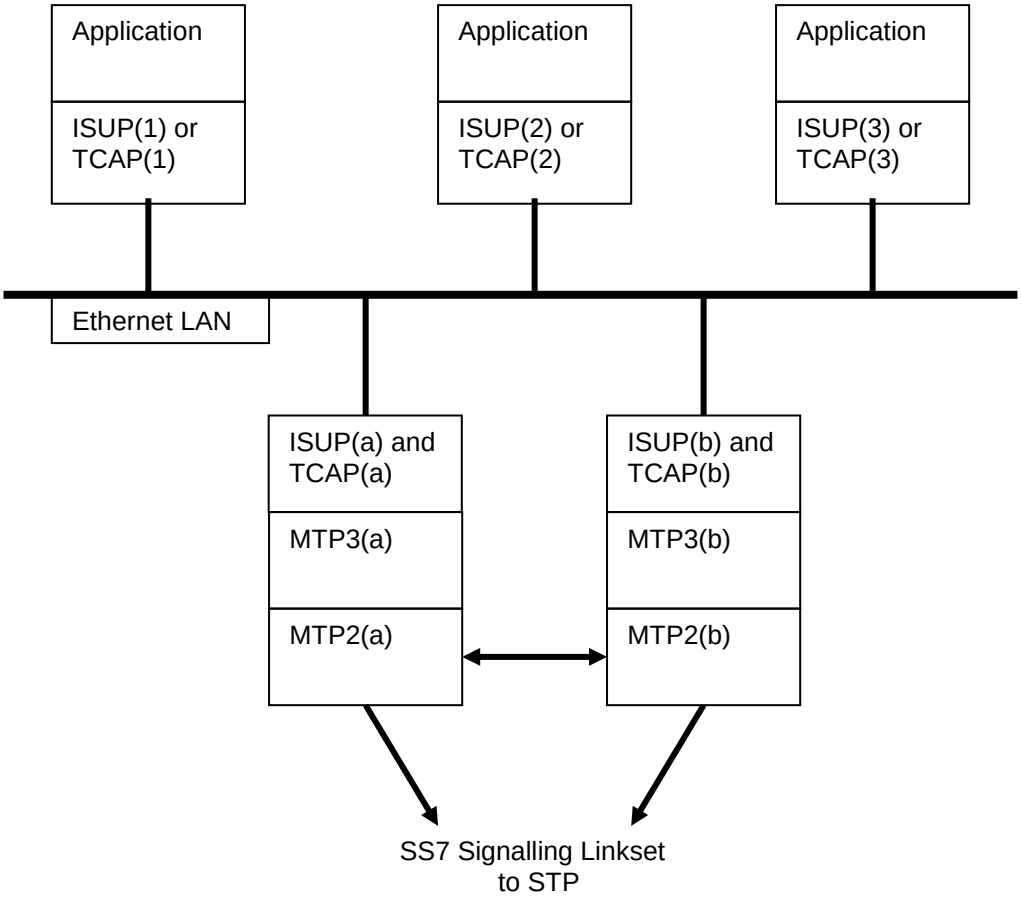
1.1 Overview

The maintenance API allows applications to obtain status from parts of the SS7 protocol stack, perform some control actions, and to receive event notifications.

The MTP3 API allows applications to implement SS7 user parts that are not supported by the Aculab SS7 protocol stack.

1.2 Aculab SS7 Architecture

The diagram below shows the components of a typical dual resilient MTP3 system with distributed ISUP and TCAP.



The maintenance API functions can be used on any of the component systems, but not all information is available from every system.

On a distributed ISUP system, ISUP information is only available for the local circuits, information about MTP3 and the signalling links is not available.

For TCAP, information is only available from the MTP3 systems and relates to the connections to the application systems.

On each MTP3 system, information is available for all ISUP circuits (provided the distributed ISUP system is connected to that MTP3 system), information is available from MTP3 and the signalling links from both the MTP3 systems.

Event indications generated by MTP3 are reported by both MTP3 systems. Event indications generated by ISUP are reported on the ISUP system and on both MTP3 systems.

1.3 Library data structures

The SS7 maintenance API library defines the following major data structures:

- An SS7 maintenance context structure used by the library as a reference into the SS7 driver. In multi-threaded programs, each thread must use a separate context structure.
- A search structure used for making status requests and request actions in the SS7 driver.
- A result structure that is used to store the results from a status request.
- An event structure. This is used by the library to return event information.

All the fields of the context structure are private to the library.

Note **All the above structures must be allocated by the functions provided by the library.**

1.4 Functional Overview

1.4.1 SS7 Status Requests and Action Requests

An application needs to perform the following steps to make a status request or perform an action in the SS7 driver.

- Using the functions from the library, create an ss7maintapi context structure. This also opens the SS7 driver.
- Create a search structure.
- For status requests, create a result structure for the library to store the results in.
- Assign the fields in the search structure and make the request.
- Check the return code from the request. If it is `ACU_SS7MAINTAPI_ERROR_SUCCESS`, then the request succeeded.
- The search and result structures can be re-used to make further requests.

1.4.2 SS7 Events

When certain states of an SS7 protocol entity (e.g. an ISUP circuit, or an MTP signalling link) change, an event is created and queued for those applications that have previously registered to retrieve that event. The events are similar to SNMP alerts, but there is no support for generating SNMP alerts from them.

See B.5 for a list of events.

Each event contains the time the protocol entity structure was created (typically the time of the firmware download or ss7maint start command), the time the event was generated, the number of times the relevant status has changed since it was created, and the new state. This information can be used to determine whether a change is happening repeatedly (e.g. a signalling link having repeated link failures).

Multiple events for the same entity (i.e. those containing repeated state changes of exactly the same item) may be suppressed (the oldest being deleted) in order to limit the size of event queue and the corresponding system memory usage.

To read events, the application must:

- Create an ss7maintapi context structure.
- Register for the required events.
- Create an event structure.
- Wait for an event to be queued.
- Read the event into the event structure.

The same context structure can be used for status and action requests.

1.4.3 MTP3 user part functions

To use the MTP3 user part API, an application needs to:

- Create an ss7maintapi context structure.
- Attach to the required local point code and service indicator.

The application can then send and receive raw MTP3 messages.

2 API Functions

2.1 Compatibility

The API library is designed so that applications do not need to be recompiled to run with a newer version of the library. However there are times when an application needs to determine whether the library supports a specific feature.

2.1.1 acu_ss7maintapi_feature_check

```
unsigned int acu_ss7maintapi_feature_check(acu_ss7maintapi_feature_t
*feature);
```

Purpose

This function checks whether the library supports the requested feature.

Parameters

`feature` Feature to check, one of:

`ACU_SS7MAINTAPI_STATS`	`amr_stats` valid in result structure
`ACU_SS7MAINTAPI_IPv6`	`ams_sockaddr`, `amr_sockaddr`, `amr_sockaddr_len`, `ame_system_ip` and `ame_remote_ip` valid.

If `ACU_SS7MAINTAPI_IPv6` is not supported then the IPv4 address is in fields `ams_ip_addr`, `amr_ip_addr` and `ame_ip_addr`. These fields are still filled for compatibility with old programs.

Return value

1 if the feature is supported, 0 if not.

2.2 Creating and deleting structures

2.2.1 acu_ss7maintapi_search_alloc

```
int acu_ss7maintapi_search_alloc(acu_ss7maintapi_search_t **search);
```

Purpose

This function allocates a new search structure for the status requests and initialises the fields so that it matches all entities.

The following fields of the search structure are common to all the requests:

`ams_next`	If non-zero, continue search from last found object **(note 1)**.

`ams_system`	System to search for object, one of:
	`ACU_SS7MAINTAPI_SYSTEM_ANY`
	`ACU_SS7MAINTAPI_SYSTEM_LOCAL`
	`ACU_SS7MAINTAPI_SYSTEM_DUAL`
	`ACU_SS7MAINTAPI_SYSTEM_DISTRIB` **(note 2)**.

`ams_localpc`	Local pointcode.
`ams_localpc_inst`	Local pointcode instance.

These fields are used by some requests; refer to the request definition for details:

`ams_remotepc`	Remote pointcode.
`ams_adjacentpc`	Pointcode of the adjacent signalling point.
`ams_slc`	Signalling link code.
`ams_cic`	ISUP circuit identity code.
`ams_cic_lim`	ISUP circuit identity code range limit.
`ams_name`	Name of M3UA client or server section (wildcard if `ams_name[0]` zero).
`ams_sockaddr`	IPv6 or IPv4 address of connected system (wildcard if `sin6_family` zero).
`ams_ssn`	SCCP Subsystem number.
`ams_tran_id`	TCAP transaction id.
`ams_route_ctx`	Routing context for M3UA connection.

Note (1) Applicable for status requests only.
Note (2) Applicable for ISUP circuit status and action requests.

Fields that are not applicable to the type of search are ignored.

The structure is initialised with all the fields set to their wildcard value (typically -1) so that all items match.

An initial status request should be made with `ams_next` set to zero. If any of the applicable fields for the request are a wild card, then the next matching object can be obtained by setting `ams_next` to a non-zero value and leaving the remainder of the search unchanged.

Parameters

`search` Search structure.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.2.2 acu_ss7maintapi_result_alloc

```
int acu_ss7maintapi_result_alloc(acu_ss7maintapi_result_t **result);
```

Purpose

This function allocates a new result structure for the status requests.

The following fields of the result structure are filled in by all the status requests:

`amr_system`	System that the information was returned from, one of:
	`ACU_SS7MAINTAPI_SYSTEM_LOCAL`
	`ACU_SS7MAINTAPI_SYSTEM_DUAL`
	The IPv4 address of a distributed ISUP system

`amr_localpc`	Local pointcode.
`amr_localpc_inst`	Local pointcode instance.

`amr_variant`	Variant of the object found, one of:
	`ACU_SS7_VARIANT_ITU`
	`ACU_SS7_VARIANT_CHINA`
	`ACU_SS7_VARIANT_ANSI`

`amr_state`	State of the found object.
`amr_reason`	Bit pattern of problems why the object is not in a normal state, or zero.
`amr_str_state`	A string version of the found state.
`amr_str_reason`	An array of reason strings followed by an empty string.

These fields are filled in by some status requests, refer to the request definition for details:

`amr_remotepc`	Remote pointcode.
`amr_adjacentpc`	Pointcode of the adjacent signalling point.
`amr_slc`	Signalling link code.
`amr_cic`	ISUP circuit identification code.
`amr_priority`	Priority of a signalling route.
`amr_trunk`	Trunk number on the card.
`amr_ts`	Timeslot.
`amr_serial`	Serial number of the card.
`amr_name`	Name of data item.
`amr_sockaddr`	IP address of connected system.
`amr_sockaddr_len`	Likely maximum field width for printing `amr_sockaddr`.
`amr_ssn`	SCCP subsystem number.
`amr_tran_id`	TCAP transaction id.
`amr_route_ctx`	M3UA Routing context.
`amr_si_mask`	Bit mask of enabled si values.
`amr_cic_registered`	ISUP CIC registration status.
`amr_stats[32]`	Statistics counts, indexed by `ACU_SS7_AMRC_xxx`.

Depending upon the type of status request being made, only certain fields from the result structure are applicable. Fields that are not applicable are usually set to `-1`.

The values returned in `amr_state` and `amr_reason` depend on the request, refer to the request definition for details.

Note **Some of the request specific fields refer to the same memory area in the buffer.**

Parameters

`result`	Result structure.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.2.3 acu_ss7maintapi_event_alloc

```
int acu_ss7maintapi_event_alloc(acu_ss7maintapi_event_t **event);
```

Purpose

This function allocates a new event buffer.

The event structure has the following user-visible fields:

ame_system	System that generated the event, one of: ACU_SS7MAINTAPI_SYSTEM_LOCAL ACU_SS7MAINTAPI_SYSTEM_DUAL ACU_SS7MAINTAPI_SYSTEM_DISTRIB The IPv4 address of the distributed ISUP system
ame_localpc	The pointcode of the local signalling point that generated the event.
ame_localpc_inst	The local pointcode instance (usually zero).
ame_server_type	The protocol stack entity that generated the event, one of the ACU_SS7MAINTAPI_SERVER_xxx values listed in section 2.6.1
ame_event_bit	A single bit identifying the event, (see B.5).
ame_remotepc	Any remote pointcode associated with the event (MTP3 and ISUP).
ame_system_ip	IP address for events from a distributed ISUP system.
ame_remote_ip	IP address of connected system (TCAP and M3UA).
ame_item	An event-dependent value, e.g. CIC number, adjacent pc, slc.
ame_item_1	Additional event dependant value.
ame_epoch	The time when the structure was created (i.e. ame_number set to 0).
ame_now	The time when the event was generated.
ame_number	The number of this event, incremented for each event.
ame_state	The new state when the event was generated.
ame_info	Event dependent information.
ame_info_1	Additional event dependant information.

The fields ame_remotepc, ame_system_ip, ame_renote_ip, ame_item and ame_item_1 identify the instance of the protocol structure to which the event relates, their exact definitions depend on the specific event generated. The fields ame_state, ame_info and ame_info_1 contain information from the protocol structure, an event is usually generated whenever the value of ame_state changes.

If any of the fields are not relevant to a specific event, then they will be set to zero.

Parameters

event	Event structure.

Return value

Zero if successful, ACU_SS7MAINTAPI_ERROR_xxx on failure.

2.2.4 acu_ss7maintapi_search_free

```
void acu_ss7maintapi_search_free(acu_ss7maintapi_search_t *search);
```

Purpose

This function deletes a search structure.

Parameters

`search` Search structure to free.

Return value

None.

2.2.5 acu_ss7maintapi_result_free

```
void acu_ss7maintapi_result_free(acu_ss7maintapi_result_t *result);
```

Purpose

This function deletes a result structure.

Parameters

`result` Result structure to free.

Return value

None.

2.2.6 acu_ss7maintapi_event_free

```
void acu_ss7maintapi_event_free(acu_ss7maintapi_event_t *event);
```

Purpose

This function deletes an event structure.

Parameters

`event` Event structure to free.

Return value

None.

2.3 Access functions

2.3.1 acu_ss7maintapi_open

```
int acu_ss7maintapi_open(acu_ss7maintapi_ctx_t **ctx);
```

Purpose

This function creates a new API context and opens the SS7 driver.

The context structure has no user-visible fields.

Note **In a multi-threaded program, each thread must use a separate context.**

Parameters

`ctx` Library context.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.3.2 acu_ss7maintapi_close

```
void acu_ss7maintapi_close(acu_ss7maintapi_ctx_t *ctx);
```

Purpose

This function closes the driver, deletes the context and any other associated resources.

Parameters

`ctx` The address of the `acu_ss7maintapi_ctx_t` structure to delete.

Return value

None.

2.3.3 acu_ss7maintapi_lib_version

```
const char *acu_ss7maintapi_lib_version(acu_ss7maintapi_ctx_t *ctx);
```

Purpose

This function returns the current version of the library.

Parameters

`ctx` Library context, may be `NULL`.

Return value

A pointer to the version string if successful, `NULL` if `ctx` specified and invalid.

2.3.4 acu_ss7maintapi_drv_version

```
const char *acu_ss7maintapi_drv_version(acu_ss7maintapi_ctx_t *ctx);
```

Purpose

This function returns the current version of the SS7 driver.

Parameters

`ctx` Library context.

Return value

A pointer to the version string if successful, `NULL` on failure.

2.4 Status functions

If wild card values were used in the search criteria, the next value can be obtained by setting the field `ams_next` in the search structure to a non-zero value and repeating the call.

The status requests can be issued from the command line by running `ss7maint apistatus`.

2.4.1 acu_ss7maintapi_status_mtp_link

```
int acu_ss7maintapi_status_mtp_link(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an SS7 signalling link.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required link.	
	`ams_remotepc`	Destination pointcode of signalling link.
	`ams_slc`	slc code of signalling link.
`result`	Result structure.	
	`amr_remotepc`	Destination pointcode of signalling link.
	`amr_slc`	slc code of signalling link.
	`amr_serial`	Serial number of the board containing the signalling link.
	`amr_trunk`	Trunk number on the board.
	`amr_ts`	Timeslot on the trunk.
	`amr_state`	Signalling link state from Appendix B.1.1
	`amr_reason`	Zero if the link is available, otherwise a bit map of problem codes from Appendix B.1.2 and Appendix B.1.3
	`amr_stats[]`	Statistic counts from MTP2. See Appendix B.1.4.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.2 acu_ss7maintapi_status_mtp_route

```
int acu_ss7maintapi_status_mtp_route(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an SS7 MTP3 route.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required route.	
	`ams_remotepc`	Pointcode of remote system.
	`ams_adjacentpc`	Pointcode of adjacent signalling point.
`result`	Result structure.	
	`amr_remotepc`	Pointcode of remote system.
	`amr_adjacentpc`	Pointcode of adjacent signalling point.
	`amr_priority`	Route priority (lower values are higher priority).
	`amr_state`	Signalling route state from Appendix B.2.1.
	`amr_reason`	Zero for a working or standby route, otherwise a bit map of problem codes from Appendix B.2.2.
	`amr_stats[]`	Statistic counts. See Appendix B.2.3

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.3 acu_ss7maintapi_status_mtp_dest

```
int acu_ss7maintapi_status_mtp_dest(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an SS7 MTP3 destination.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required destination.	
	`ams_remotepc`	Pointcode of destination.
`result`	Result structure.	
	`amr_remotepc`	Pointcode of destination.
	`amr_state`	Destination state from Appendix B.3.1.
	`amr_reason`	Zero if the destination is accessible, otherwise a bit map of values from Appendix B.3.2.
	`amr_stats[]`	Statistic counts. See Appendix B.3.3

Discrepancies between the `amr_stats[]` values and the sum of those for the related routes can occur if messages are received for which there is no corresponding route (ie when no reply can be sent).

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.4 acu_ss7maintapi_status_isup_cic

```
int acu_ss7maintapi_status_isup_cic(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an SS7 ISUP circuit.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required circuit.	
	`ams_remotepc`	Destination pointcode of ISUP link.
	`ams_cic`	Lowest numbered CIC.
	`ams_cic_lim`	Highest CIC (inclusive).
`result`	Result structure.	
	`amr_remotepc`	Destination pointcode of ISUP link.
	`amr_cic`	CIC. number
	`amr_serial`	Serial number of the board containing the CIC.
	`amr_trunk`	Trunk number on the board.
	`amr_ts`	Timeslot on the trunk.
	`amr_state`	ISUP circuit state from Appendix B.4.1.
	`amr_reason`	Zero if the connection is in a normal (and unblocked) state, otherwise a bit map of problem codes from Appendix B.4.2.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.5 acu_ss7maintapi_status_tcap_application

```
int acu_ss7maintapi_status_tcap_application(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about the connection from a TCAP application to the SS7 device driver.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required tcap application.	
	`ams_sockaddr`	IP address of TCAP application system.
	`ams_ssn`	SCCP subsystem number being used by the application.
	`ams_tran_id`	TCAP transaction.id block (masked with `0xfff00000`).
`result`	Result structure.	
	`amr_sockaddr`	IP address of TCAP application system.
	`amr_ssn`	SCCP subsystem number being used by the application.
	`amr_tran_id`	TCAP transaction.id block (masked with `0xfff00000`).
	`amr_name`	Name of application (from `trace_tag` configuration option)
	`amr_state`	Bit map of states from Appendix B.5.1.
	`amr_reason`	Always zero.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.6 acu_ss7maintapi_status_m3ua_connection

```
int acu_ss7maintapi_status_m3ua_connection(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an M3UA connection and its associated routing contexts. A separate result is returned for each routing context.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required M3UA information.	
	`ams_name`	Name of client, server or ipsp configuration section.
	`ams_sockaddr`	IP address of remote M3UA system.
	`ams_route_ctx`	Routing context number (from local configuration).
`result`	Result structure.	
	`amr_name`	Name of client, server or ipsp configuration section.
	`amr_sockaddr`	IP address of remote M3UA system.
	`amr_route_ctx`	Routing context number (from local configuration).
	`amr_state`	One of the states from Appendix B.6.2
	`amr_reason`	

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.7 acu_ss7maintapi_status_m3ua_con_stats

```
int acu_ss7maintapi_status_m3ua_con_stats(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about an M3UA connection.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required M3UA information.	
	`ams_name`	Name of client, server or ipsp configuration section.
	`ams_sockaddr`	IP address of remote M3UA system.
	`ams_route_ctx`	Routing context number (from local configuration).
`result`	Result structure.	
	`amr_name`	Name of client, server or ipsp configuration section.
	`amr_sockaddr`	IP address of remote M3UA system.
	`amr_route_ctx`	Routing context number (from local configuration).
	`amr_state`	One of the states from Appendix B.6.3
	`amr_reason`	
	`amr_stats[]`	Connection statistics. See Appendix B.6.4

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.8 acu_ss7maintapi_status_m3ua_address

```
int acu_ss7maintapi_status_m3ua_address(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about the addresses associated with an M3UA connection. It differs from `acu_ss7maintapi_status_m3ua_con()` because it returns separate information for each remote point code.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required M3UA information.	
	`ams_name`	Name of client, server or ipsp configuration section.
	`ams_sockaddr`	IP address of remote M3UA system.
	`ams_route_ctx`	Routing context number (from local configuration).
	`ams_remotepc`	SS7 pointcode of the SCCP or ISUP peer.
`result`	Result structure.	
	`amr_name`	Name of client, server or ipsp configuration section.
	`amr_sockaddr`	IP address of remote M3UA system.
	`amr_route_ctx`	Routing context number (from local configuration).
	`amr_remotepc`	SS7 pointcode of the SCCP or ISUP peer.
	`ams_si_mask`	Bitmask indicating which si values are enabled.
	`amr_state`	One of the states from Appendix B.6.2
	`amr_reason`	

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.4.9 acu_ss7maintapi_status_m3ua_route_key

```
int acu_ss7maintapi_status_m3ua_route_key(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_result_t *result);
```

Purpose

This function returns information about each route_key (across all connections).

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required M3UA information.	
	`ams_name`	Name of client, server or ipsp configuration section.
	`ams_sockaddr`	IP address of remote M3UA system.
	`ams_route_ctx`	Routing context number (from local configuration).
	`ams_remotepc`	SS7 pointcode of the SCCP or ISUP peer.
`result`	Result structure.	
	`amr_name`	Name of client, server or ipsp configuration section.
	`amr_sockaddr`	IP address of remote M3UA system.
	`amr_route_ctx`	Routing context number (from local configuration).
	`amr_remotepc`	SS7 pointcode of the SCCP or ISUP peer.
	`ams_si_mask`	Bitmask indicating which si values are enabled.
	`amr_state`	One of the states from Appendix B.6.1
	`amr_reason`	Always zero.
	`amr_stats[]`	Route key statistics, see Appendix B.6.5

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.5 Action functions

The action functions perform the requested action on all entities that match the search criterion. An error is returned if nothing matches or if an excessive number of items match.

The action functions can be issued from the command line using `ss7maint apiaction`.

2.5.1 acu_ss7maintapi_mtp_link

```
int acu_ss7maintapi_mtp_link(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_link_action_t action);
```

Purpose

This function performs an action on all SS7 signalling links that match the search criteria.

A maximum of 10000 signalling links can have an action performed on them before the error `ACU_SS7MAINTAPI_ERROR_TOO_MANY` is returned.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required signalling links.	
	`ams_remotepc`	Destination pointcode of signalling link.
	`ams_slc`	slc code of signalling link.
`action`	Action to perform, one of:	
	`ACU_SS7_MTP_LINK_INHIBIT`	Inhibit the links.
	`ACU_SS7_MTP_LINK_UNINHIBIT`	Uninhibit the links.
	`ACU_SS7_MTP_LINK_ACTIVATE`	Activate the links.
	`ACU_SS7_MTP_LINK_DEACTIVATE`	Deactivate the links.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.5.2 acu_ss7maintapi_isup_cic

```
int acu_ss7maintapi_isup_cic(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_search_t *search, acu_ss7maintapi_cic_action_t action);
```

Purpose

This function performs an action on all the ISUP circuits that match the search criteria.

A maximum of 10000 trunks can have an action performed on them before the error `ACU_SS7MAINTAPI_ERROR_TOO_MANY` is returned.

Parameters

`ctx`	Library context for this request.	
`search`	Search details identifying required circuits.	
	`ams_remotepc`	Destination pointcode of ISUP link.
	`ams_cic`	Lowest numbered CIC.
	`ams_cic_lim`	Highest CIC (inclusive).
`action`	Action to perform, one of:	
	`ACU_SS7_ISUP_CIC_BLOCK`	Maintenance block the circuits.
	`ACU_SS7_ISUP_CIC_UNBLOCK`	Maintenance unblock the circuits.
	`ACU_SS7_ISUP_CIC_RESET`	Reset the circuits.
	`ACU_SS7_ISUP_CIC_HW_BLOCK`	Hardware block the circuits.
	`ACU_SS7_ISUP_CIC_HW_UNBLOCK`	Hardware unblock the circuits.

If the search matches on more than one circuit of a particular trunk, ISUP will send group blocking, group unblocking or group reset messages.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.5.3 acu_ss7maintapi_tcap

No actions are currently defined for TCAP.

2.5.4 acu_ss7maintapi_m3ua

```
int acu_ss7maintapi_m3ua(acu_ss7maintapi_ctx_t *ctx, acu_ss7maintapi_search_t
*search, acu_ss7maintapi_m3ua_action_t action);
```

Purpose

This function performs an action on all the M3UA routing keys that match the search criteria.

A maximum of 10000 keys can have an action performed on them before the error `ACU_SS7MAINTAPI_ERROR_TOO_MANY` is returned.

Parameters

`ctx` Library context for this request.

`search` Search details identifying required routing keys.

- `ams_name` Name of client, server or ipsp configuration section.
- `ams_sockaddr` IP address of remote M3UA system.
- `ams_route_ctx` Routing context number (from local configuration).

`action` Action to perform, one of:

- `ACU_SS7_M3UA_CONNECT` Connect to remote system.
- `ACU_SS7_M3UA_DISCONNECT` Disconnect from remote system.
- `ACU_SS7_M3UA_RK_REGISTER` Register route key.
- `ACU_SS7_M3UA_RK_DEREGISTER` Deregister route key.
- `ACU_SS7_M3UA_RK_ACTIVATE` Activate route key.
- `ACU_SS7_M3UA_RK_INACTIVATE` Inactivate route key.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.6 Event functions

The event functions can be issued from the command line by running `ss7maint apievent`.

2.6.1 acu_ss7maintapi_event_register

```
int acu_ss7maintapi_event_register(acu_ss7maintapi_ctx_t *ctx, int localpc,
int localpc_inst, int server, unsigned int event_bits);
```

Purpose

This function enables events from the specified protocol stack entities.

Once enabled, events are queued within the SS7 driver until read by the application. Old events for the same item may be deleted.

Events can be enabled before the protocol stack is started.

If called multiple times for the same `ctx`, the effects are additive.

There is a separate event queue for each library `ctx`. A single event can be queued for multiple contexts.

Parameters

`ctx`	Library context for this request.
`localpc`	Local pointcode from which events are required, or `-1` for all pointcodes.
`localpc_inst`	Instance of local pointcode, or `-1` for all instances.
`server`	Protocol stack entity from which events are required, one of:
`ACU_SS7MAINTAPI_SERVER_ANY`	Events from all servers.
`ACU_SS7MAINTAPI_SERVER_MTP3`	Events from MTP3
`ACU_SS7MAINTAPI_SERVER_ISUP`	Events from ISUP
`ACU_SS7MAINTAPI_SERVER_DUAL_LINK`	Events from the inter-dual link
`ACU_SS7MAINTAPI_SERVER_SCCP`	Events from SCCP
`ACU_SS7MAINTAPI_SERVER_TCAP`	Events from TCAP
`ACU_SS7MAINTAPI_SERVER_M3UA`	Events from SIGTRAN M3UA
`event_bits`	Bit pattern of events to enable.

See B.5 for the defined events for each server.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.6.2 acu_ss7maintapi_event_deregister

```
int acu_ss7maintapi_event_deregister(acu_ss7maintapi_ctx_t *ctx, int localpc,
int localpc_inst, int server, unsigned int event_bits);
```

Purpose

This function disables the specified events.

Parameters

`ctx`	Library context for this request.
`localpc`	Local pointcode from which events are required, or `-1` for all pointcodes.
`localpc_inst`	Instance of local pointcode, or `-1` for all instances.
`server`	Protocol stack entity from which events are required.
`event_bits`	Bit pattern of events to disable.

Note **Events already queued are not affected.**

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.6.3 acu_ss7maintapi_event_get

```
int acu_ss7maintapi_event_get(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_event_t *event, int tmo_ms);
```

Purpose

This function reads an event from the driver.

See section 2.2.3 for details of the event structure, and Appendix C: for the definition of the events.

Parameters

`ctx`	Library context for this request.
`event`	Event structure to fill with the event information.
`tmo_ms`	Time to wait in milliseconds, `0` => don't wait, `-1` => wait forever.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.6.4 acu_ss7maintapi_event_get_os_event

```
int acu_ss7maintapi_event_get_os_event(acu_ss7maintapi_ctx_t *ctx,
acu_ss7maintapi_os_event_t *os_event);
```

Purpose

This function obtains an operating-system specific token that can be used to wait for events to be queued.

On Windows it obtains the `HANDLE` of a windows event that can be used in `WaitForSingleObject()` or `WaitForMultipleObjects()`.

On Linux it obtains the underlying fd number that can be used in `poll()` or `select()`.

Parameters

`ctx`	Library context for this request
`os_event`	Operating-system specific event value.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.7 MTP3 user part functions

These functions are included in the maintenance API library because they use the same basic method to access the SS7 driver. It is also likely that any application will need to use the 'maintenance' functions as well as those described here.

Although described as an MTP3 API, the traffic may be carried by M3UA, however the message size limit for MTP always applies.

The three C structures defined in this section have padding fields so that future expansion will not need to change the sizes. These structures can be safely allocated by the application (unlike the other structures in the maintenance API).

The send and receive functions can be blocking or non-blocking. The event / fd number obtained by `acu_ss7maintapi_event_get_os_event()` can be used to determine when receive data is available or send flow control restrictions are lifted.

2.7.1 acu_ss7maintapi_mtp3_attach

```
int acu_ss7maintapi_mtp3_attach(acu_ss7maintapi_ctx_t *ctx, int flags,
acu_ss7maintapi_mtp3_attach_info_t *attach_info);
```

Purpose

This function attaches to MTP3 for the specified user part.

Parameters

`ctx`	Library context for this request.	
`flags`	Bitwise 'or' of:	
	`ACU_SS7MAINTAPI_TX_ONLY`	Attach to transmit only
`attach_info`	Defines the protocol entity and user part:	
	`am3a_localpc`	Local point code.
	`am3a_localpc_inst`	Local point code instance (usually 0)
	`am3a_ni`	Network Indicator
	`am3a_si`	Service indicator (2 to 15)
	`am3a_variant`	SS7 protocol variant (obtained from the driver).
	`am3a_maxsu`	Maximum user part data bytes.

Only a single context can attach to receive traffic, multiple contexts can send traffic.

The network indicator can be set to `ACU_SS7MAINTAPI_ADOPT_NI` to request the value configured in MTP3, `ACU_SS7MAINTAPI_WILD_NI` to request traffic for all network indicators, or 0 through 3 for a specific network indicator. If `ACU_SS7MAINTAPI_ADOPT_NI` is requested then `am3a_ni` is modified to be the actual network indicator in use.

Note **For compatibility with future releases the entire `acu_ss7maintapi_mtp3_attach_info_t` structure should be zeroed prior to setting the required fields.**

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.7.2 acu_ss7maintapi_mtp3_detach

```
void acu_ss7maintapi_mtp3_detach(acu_ss7maintapi_ctx_t *ctx);
```

Purpose

This function detaches the context from MTP3.

Parameters

`ctx`	Library context for this request.

Detaching the context will also unblock an `acu_ss7maintapi_mtp3_recv()` call that is waiting for data. On windows it may be called from a function passed to `SetConsoleCtrlHandler()` in order to unblock when Ctrl-C is typed.

It is not necessary to detach prior to calling `acu_ss7maintapi_close()` or program exit.

Note **This is the only function in this library that can be called from a 2nd thread specifying the same ctx.**

2.7.3 acu_ss7maintapi_mtp3_send

```
int acu_ss7maintapi_mtp3_send(acu_ss7maintapi_ctx_t *ctx, int flags, const
acu_ss7maintapi_mtp3_send_info_t *send_info, const void *buffer, int msglen);
```

Purpose

This function sends the specified data.

Parameters

`ctx`	Library context for this request.	
`flags`	Bitwise 'or' of:	
	`ACU_SS7MAINTAPI_NDELAY`	Do not block waiting for buffers.
	`ACU_SS7MAINTAPI_TO_DUAL`	Send to peer of MTP3 dual system.
	`ACU_SS7MAINTAPI_FROM_DUAL`	Send from peer of MTP3 dual system.
`send_info`	Defines the MTP3 routing label:	
	`am3s_remotepc`	Remote point code.
	`am3s_sls`	Signalling link selector.
	`am3s_msg_pri`	Message priority (ANSI MTP).
`buffer`	User part data to send	
`msglen`	Length of user part data (bytes)	

Note **For compatibility with future releases the entire `acu_ss7maintapi_mtp3_send_info_t` structure should be zeroed prior to setting the required fields.**

If the message cannot be sent (because all of the transmit buffers assigned to the user are busy) then if `ACU_SS7MAINTAPI_NDELAY` is specified the call will return `ACU_SS7MAINTAPI_ERROR_CONGESTION` otherwise it will block (interruptible on Linux systems).

Setting `ACU_SS7MAINTAPI_TO_DUAL` will cause the message to be sent to the dual (of a dual MTP3 system) as if received from `am3s_remotepc`. This can be used to transfer a message to the peer application (running on the other system of the dual pair) when a message contains a local reference that is locally unknown.

Note **Ensure that the message hasn't come from the dual peer first otherwise the message is likely to be reflected forever.**

Setting `ACU_SS7MAINTAPI_FROM_DUAL` will cause to the message to be passed over the dual link before being processed by MTP3 routing. This may be useful for conformance testing of load balancing.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

2.7.4 acu_ss7maintapi_mtp3_recv

```
int acu_ss7maintapi_mtp3_recv(acu_ss7maintapi_ctx_t *ctx, int flags,
acu_ss7maintapi_mtp3_recv_info_t *recv_info, const void *buffer, int buflen);
```

Purpose

This function receives an event from MTP3.

Parameters

`ctx`	Library context for this request.	
`flags`	Bitwise 'or' of:	
	`ACU_SS7MAINTAPI_NDELAY`	Do not block waiting for an event
`recv_info`	Describes the returned event:	
	`am3r_indication`	Event type, one of:
	`ACU_SS7_MTP3_IND_DATA`	Data from remote system.
	`ACU_SS7_MTP3_IND_PAUSE`	Remote system inaccessible.
	`ACU_SS7_MTP3_IND_RESUME`	Remote system accessible.
	`ACU_SS7_MTP3_IND_CONGESTION`	Remote system congested.
	`ACU_SS7_MTP3_IND_STATUS`	User part unavailable (UPU).
	`am3r_remotepc`	Remote point code.
	`am3r_adjacentpc`	Adjacent point code.
	`am3r_ni`	Received network indicator.
	`am3r_sls`	Signalling link selector.
	`am3r_msg_pri`	Message priority (ANSI MTP).
	`am3r_msg_len`	Length of received user part data.
	`am3r_cong_lvl`	Received congestion level.
	`am3r_up_cause`	Received UPU cause.
`buffer`	Buffer for receive user part data	
`buflen`	Length of buffer (bytes)	

If no events are available then if `ACU_SS7MAINTAPI_NDELAY` is specified the call will return `ACU_SS7MAINTAPI_ERROR_NOTFOUND` otherwise it will block (interruptible on Linux systems).

If the supplied buffer isn't long enough, then `ACU_SS7MAINTAPI_ERROR_TOO_LONG` is returned and the message discarded.

If `am3r_adjacentpc` is set to the local point code, then the message has been transferred from the other system of a dual pair (either by the driver because no user part was registered, or by the user part code itself). The application must not send such messages back to the dual peer as they are likely to be transferred between the systems for ever.

Return value

Zero if successful, `ACU_SS7MAINTAPI_ERROR_xxx` on failure.

Appendix A: Building applications

Linux

The API header file includes all the necessary system headers.
Compile with `-D_REENTRANT`.
Link with `-L ${ACULAB_ROOT}/lib -lacu_ss7maintapi`.
You may also want to specify `-Wl,-rpath,${ACULAB_ROOT}/lib -Wl,--enable-new-dtags` so that the library is found at run time.

For 64bit applications replace `lib` with `lib64`.

Windows

Applications must be compiled as threaded programs. i.e. built with `-MT` (or `-MTd`) not `-ML`.
The library filename is `acuss7_maintapi.lib`.

Appendix B: SS7 States, Problem Codes and Statistics

Note **The 'problem codes' are bit significant, more than one bit may be set, and additional bits may be set by future software releases.**

B.1 MTP Signalling Links

B.1.1 MTP Signalling Link States

ACU_SS7_MTP_LINK_L3L2_AVAILABLE	Normal state. The signalling link is available for use at both MTP level 3 and MTP level 2.
ACU_SS7_MTP_LINK_L3_UNAVAILABLE	Abnormal state. The signalling link is not available at MTP level 3.
ACU_SS7_MTP_LINK_L2_UNAVAILABLE	Abnormal transitory state. The signalling link is not available at MTP level 2.
ACU_SS7_MTP_LINK_L3L2_UNAVAILABLE	Abnormal state. The signalling link is not available at MTP level 3 or MTP level 2.

B.1.2 MTP Level 2 Signalling Link Problem Codes

ACU_SS7_MTP_LINK_P_L2_UNKNOWN	Either the driver cannot obtain the MTP level 2 link state, or the state as reported by the driver is not known (denoting a driver ss7maint api version mis-match).
ACU_SS7_MTP_LINK_P_L2_PWR_OFF	The link is powered off.
ACU_SS7_MTP_LINK_P_L2_OUT_OF_SRV	The link is out of service and sending SIOS.
ACU_SS7_MTP_LINK_P_L2_ALIGNED_RDY	The link is aligned waiting for the first FISU or MSU before going into service.
ACU_SS7_MTP_LINK_P_L2_ALIGNED_N_RDY	The link is aligned but cannot go into service due to processor outage.
ACU_SS7_MTP_LINK_P_L2_NOT_ALIGNED	Initial alignment has started. SIO has been sent and timer T2 started.
ACU_SS7_MTP_LINK_P_L2_ALIGNED	SIO has been received from the remote end. SIN/SIE has been sent and timer T3 started.
ACU_SS7_MTP_LINK_P_L2_PROVING	SIN/SIE has been received and the link is in the proving stage of initial alignment. Timer T4 has been started.
ACU_SS7_MTP_LINK_P_L2_LOC_BLOCKED	The link is locally blocked.
ACU_SS7_MTP_LINK_P_L2_REM_BLOCKED	The link is remotely blocked (RPO received).

B.1.3 MTP Level 3 Signalling Link Problem Codes

ACU_SS7_MTP_LINK_P_L3_NO_CONFIG	The link is not configured. Check configuration data and ensure the SS7 driver has been started with ss7maint start.
ACU_SS7_MTP_LINK_P_L3_UNKNOWN	The MTP level 3 link state as reported by the driver is not known. Denotes a driver ss7maint api version mis-match.
ACU_SS7_MTP_LINK_P_L3_INACTIVE	The link is currently inactive.
ACU_SS7_MTP_LINK_P_L3_DEACTIVATED	The link has been manually deactivated.
ACU_SS7_MTP_LINK_P_L3_MTP2_FAIL	MTP level 2 has reported a failure to MTP level 3.
ACU_SS7_MTP_LINK_P_L3_SLT_FAIL	SLTM / SLTA failure. Check pointcodes and signalling link codes in the configuration data.
ACU_SS7_MTP_LINK_P_L3_ACTIVATING	MTP level 3 is in the process of trying to activate the link.
ACU_SS7_MTP_LINK_P_L3_LOC_INHIB	The link has been locally inhibited.
ACU_SS7_MTP_LINK_P_L3_REM_INHIB	The link has been remotely inhibited.
ACU_SS7_MTP_LINK_P_L3_REM_BLOCKED	The link is remotely blocked (RPO received).

B.1.4 MTP Level 2 Signalling Link Statistics

ACU_SS7_AMRC_TIME	Time since link created.
ACU_SS7_AMRC_TX_MSGS	Number of transmitted msus.
ACU_SS7_AMRC_TX_BYTES	msu bytes (including header, crc and one flag).
ACU_SS7_AMRC_RETX_MSGS	Retransmitted msus.
ACU_SS7_AMRC_RETX_BYTES	Retansmitted msu bytes.
ACU_SS7_AMRC_RX_MSGS	Received (in sequence) msus.
ACU_SS7_AMRC_RX_BYTES	Bytes of received msu.
ACU_SS7_AMRC_RX_DISCARDS	Receive msus discarded
ACU_SS7_AMRC_RX_DISCARD_BYTES	Bytes of discarded msus
ACU_SS7_AMRC_ERRORS	Number of receive errors.
ACU_SS7_AMRC_LAST_ERROR	Type of last receive error.
ACU_SS7_AMRC_ERR_TIME	Time since last receive error.
ACU_SS7_AMRC_FAILURES	Number of link failures.
ACU_SS7_AMRC_LAST_FAIL	Type of last link failure.
ACU_SS7_AMRC_FAIL_TIME	Time since last link failure.

The least significant 8 bits of the ‘last receive error’ and ‘last link failure’ values is the link state, and is one of:

0x0000	Idle	Sending SIOS.
0x0001	Not Aligned	Sending SIO waiting for SIO.
0x0002	Aligned	Sending SIN/SIE waiting for SIN/SIE.
0x0003	Proving	Sending and receiving SIN.
0x0004	Emergency proving	Sending and receiving SIE.
0x0005	Aligned ready	Sending FISU waiting for FISU.
0x0006	Data transfer	Link up

The higher bits of the ‘last receive error’ value are one of:

0x0100	HDLC abort received
0x0200	Receive frame too long (sif greater than 272 bytes).
0x0300	Receive frame not a multiple of 8 bits.
0x0400	Receive CRC error.
0x0500	Receive frame too short.
0x0600	LI field incorrect for frame length.
0x0700	No receive buffer available.
0x1100	First msu with invalid BSN
0x1200	Frame following bad BSN.
0x1300	First msu with invalid FIB.
0x1400	Frame following bad FIB.
0x1500	Incorrect FIB received, waiting retransmission.
0x1600	Not in data transfer.
0x1700	Bad FSN, retransmit requested.
0x1800	Duplicated FSN.
0x1900	Congestion discard.

The higher bits of the ‘last link failure’ value are one of:

0x8000	User (ie mtp3) requested disconnect.
0x8100	Protocol timer expired.
0x8200	LSSU invalid for current state.
0x8300	Invalid BSN received.
0x8400	Invalid FIB received.
0x8500	Error rate monitor (check layer 1 framing).

The HDLC abort, not multiple of 8 bits, and CRC receive errors are most likely caused by TDM clock slip.

B.2 MTP Signalling Routes

B.2.1 MTP Signalling Route States

ACU_SS7_MTP_ROUTE_WORKING	Normal state. The route is used to carry traffic to the destination.
ACU_SS7_MTP_ROUTE_STANDBY	Normal state. Higher priority Working or Working / Restricted routes exist to the destination.
ACU_SS7_MTP_ROUTE_WORKINGRES	Abnormal state. The route is used to carry traffic to the destination but is restricted. A TFR message has been received.
ACU_SS7_MTP_ROUTE_STANDBYRES	Abnormal state. Equal priority Working or higher priority Working or Working / Restricted routes exist to the destination.
ACU_SS7_MTP_ROUTE_PROHIBITED	Abnormal state. A TFP message has been received for this route.
ACU_SS7_MTP_ROUTE_UNAVAILABLE	Abnormal state. The linkset over which the route is defined is unavailable.
ACU_SS7_MTP_ROUTE_UNAVAILRES	Abnormal state. The linkset over which the route is defined is unavailable. The route will recover to Working / Restricted or Standby / Restricted on linkset recovery.
ACU_SS7_MTP_ROUTE_UNAVAILPRO	Abnormal state. The linkset over which the route is defined is unavailable. The route will recover to Prohibited on linkset recovery.
ACU_SS7_MTP_ROUTE_UNKNOWN	Abnormal state. The route state as reported by the driver is not known. Denotes a driver ss7maint api version mis-match.

B.2.2 MTP Signalling Route Problem Codes

ACU_SS7_MTP_ROUTE_P_TFR_RECEIVED	A Transfer Restricted (TFR) message has been received for this route.
ACU_SS7_MTP_ROUTE_P_TFP_RECEIVED	A Transfer Prohibited (TFP) message has been received for this route.
ACU_SS7_MTP_ROUTE_P_LINKSET_FAIL	The linkset over which this route is defined is unavailable.
ACU_SS7_MTP_ROUTE_P_MTP_RESTART	The route is unavailable because MTP restart is ongoing in the local signalling point or the adjacent signalling point.

B.2.3 MTP Signalling Route Statistics

ACU_SS7_AMRC_TX_MSGS	Number of transmitted pdus.
ACU_SS7_AMRC_TX_BYTES	Transmitted bytes (including routing label).
ACU_SS7_AMRC_RX_MSGS	Received pdus.
ACU_SS7_AMRC_RX_BYTES	Bytes of received pdu.

B.3 MTP Destinations

B.3.1 MTP Destination States

ACU_SS7_MTP_DEST_ACCESSIBLE	Normal state. The destination can be reached and traffic can be sent to and received from the destination.
ACU_SS7_MTP_DEST_RESTRICTED	Abnormal state. The destination can be reached, but only by using restricted routes.
ACU_SS7_MTP_DEST_INACCESSIBLE	Abnormal state. The destination cannot be reached.
ACU_SS7_MTP_DEST_UNKNOWN	Abnormal state. The destination state as reported by the driver is not known. Denotes a driver ss7maint api version mis-match.

B.3.2 MTP Destination Problem Codes

ACU_SS7_MTP_DEST_P_OWNSP_RESTART	MTP restart is in progress in the local signalling point.
ACU_SS7_MTP_DEST_P_ADJ_RESTART	The adjacent signalling point is currently restarting.
ACU_SS7_MTP_DEST_P_ROUTE_RESTR	The destination can be reached, but only using restricted routes.
ACU_SS7_MTP_DEST_P_ROUTE_FAILURE	All routes to this destination have failed.

B.3.3 MTP Destination Statistics:

ACU_SS7_AMRC_TIME	Time since destination created.
ACU_SS7_AMRC_TX_MSGS	Number of transmitted pdus.
ACU_SS7_AMRC_TX_BYTES	Transmit bytes (including routing label).
ACU_SS7_AMRC_TX_DISCARDS	Discards because no route.
ACU_SS7_AMRC_RX_MSGS	Received pdus.
ACU_SS7_AMRC_RX_BYTES	Bytes of received pdu.

B.4 ISUP Circuits

B.4.1 ISUP Circuit States

ACU_SS7_ISUP_CIC_IDLE	Normal state. The circuit is idle waiting for an incoming or outgoing call to be made.
ACU_SS7_ISUP_CIC_INCOMING	Normal state. An incoming call is being made.
ACU_SS7_ISUP_CIC_OUTGOING	Normal state. An outgoing call is being made.
ACU_SS7_ISUP_CIC_CONNECTED	Normal state. The call is in a stable connected state.
ACU_SS7_ISUP_CIC_RELEASED	Normal state. A call using this circuit is being released.
ACU_SS7_ISUP_CIC_FREE	Normal transitory state. The call has been released and resources associated with this call are waiting to be freed.
ACU_SS7_ISUP_CIC_UNKNOWN	Abnormal state. The circuit state as reported by the driver is not known. Denotes a driver ss7maint api version mis-match.
ACU_SS7_ISUP_CIC_FAIL_IDLE	Abnormal state. These states denote the call is in one of the above call states, but the circuit has failed. One or more of the ISUP circuit problem code bits from Appendix B.4.2 will be set.
ACU_SS7_ISUP_CIC_FAIL_INCOMING	
ACU_SS7_ISUP_CIC_FAIL_OUTGOING	
ACU_SS7_ISUP_CIC_FAIL_CONNECTED	
ACU_SS7_ISUP_CIC_FAIL_RELEASED	
ACU_SS7_ISUP_CIC_FAIL_FREE	
ACU_SS7_ISUP_CIC_FAIL	Bit set to differentiate the ‘fail’ states from the ‘normal’ states.

B.4.2 ISUP Circuit Problem Codes

ACU_SS7_ISUP_CIC_P_LOC_MAINTENANCE	The circuit has been locally maintenance blocked.
ACU_SS7_ISUP_CIC_P_LOC_HARDWARE	The circuit has been locally hardware blocked.
ACU_SS7_ISUP_CIC_P_REM_MAINTENANCE	The circuit has been remotely maintenance blocked.
ACU_SS7_ISUP_CIC_P_REM_HARDWARE	The circuit has been remotely hardware blocked.
ACU_SS7_ISUP_CIC_P_UCIC	An unequipped circuit identification code (UCIC) message has been received.
ACU_SS7_ISUP_CIC_P_UNBLOCK_PENDING	An outward unblock is in progress.
ACU_SS7_ISUP_CIC_P_BLOCK_PENDING	An outward block is in progress.
ACU_SS7_ISUP_CIC_P_INWARD_RESET	An inward reset is waiting for an application to respond to an inward clear.
ACU_SS7_ISUP_CIC_P_RESET_PENDING	An outward reset is in progress
ACU_SS7_ISUP_CIC_P_ISUP_RESTART	Optional depending upon configuration. An MTP-RESUME primitive has been received. An ISUP User-Part Test (UPT) message has been sent. Local circuits remain unavailable until a response is received from the remote ISUP.
ACU_SS7_ISUP_CIC_P_UP_INACCESSIBLE	An MTP-STATUS primitive has been received with cause inaccessible remote user.
ACU_SS7_ISUP_CIC_P_UP_UNEQUIPPED	An MTP-STATUS primitive has been received with cause unequipped remote user.
ACU_SS7_ISUP_CIC_P_UP_UNAVAILABLE	An MTP-STATUS primitive has been received with cause unknown.
ACU_SS7_ISUP_CIC_P_MTP_PAUSED	An MTP-PAUSE primitive has been received.
ACU_SS7_ISUP_CIC_P_MTP_UNAVAILABLE	Distributed ISUP not connected to local MTP3.

B.5 TCAP

B.5.1 TCAP application states

This is the state of the connection to the tcap application from the driver's point of view. The field is bit-significant.

ACU_SS7_EVS_TCAP_REGISTERED	TCAP application has created an ssap.
ACU_SS7_EVS_TCAP_SERVER	Application has 'server' enabled.
ACU_SS7_EVS_TCAP_UNI_SERVER	Application has 'uni_server' enabled.
ACU_SS7_EVS_TCAP_ANSI	ANSI TCAP in use.
ACU_SS7_EVS_TCAP_ALT_REG	Another ssap is registered for this ssn.
ACU_SS7_EVS_TCAP_ALT_SVR	Another 'server' exists for this ssn.
ACU_SS7_EVS_TCAP_ALT_UNI_SVR	Another 'uni_server' exists for this ssn.

B.6 SIGTRAN M3UA

B.6.1 M3UA application server states

This is the overall state of an application server (i.e. of a routing context or key). The value is that of the 'Status type' and 'Status Information' field of the last M3UA management notify message sent by the system (for clients the notify message isn't sent, but the logic is the same).

ACU_SS7_M3UA_AS_INACTIVE	Not carrying traffic.
ACU_SS7_M3UA_AS_ACTIVE	Carrying Traffic.
ACU_SS7_M3UA_AS_PENDING	Traffic being queued waiting key activation.
ACU_SS7_M3UA_INSUFFICIENT_ASP	Insufficient active connections.

Additionally the following values are reported in events, however they don't affect the reported state.

ACU_SS7_M3UA_ALT_ASP_ACTIVE	Another connection is now taking the traffic.
ACU_SS7_M3UA_ASP_FAILURE	Some ASP has failed.

B.6.2 M3UA connection routing context states

This is the state an M3UA routing context for a specific SCTP connection.

ACU_SS7_M3UA_CRK_DISCONNECTED	SCTP connection disconnected **(note 1)**.
ACU_SS7_M3UA_CRK_IDLE	Connected but not registered.
ACU_SS7_M3UA_CRK_REGISTERING	Registration in progress.
ACU_SS7_M3UA_CRK_REGISTERED	Registered but not activated.
ACU_SS7_M3UA_CRK_ACTIVATING	Activation in progress.
ACU_SS7_M3UA_CRK_ACTIVATED	Activated, waiting NOTIFY(AS-ACTIVE).
ACU_SS7_M3UA_CRK_ACTIVE	Carrying traffic.
ACU_SS7_M3UA_CRK_INACTIVATING	Inactivation in progress.
ACU_SS7_M3UA_CRK_DEREGISTERING	Deregistration in progress.

Note (1) Only reported in events for client and ipsp_clients.

B.6.3 M3UA connection states

This is the state of an M3UA SCTP connection.

CS_IDLE	Disconnected, connect not required.
CS_LISTEN	Listening socket, should not be returned.
CS_NEED_CONNECT	Connect requested (transient state).
CS_CONNECT_RETRY	Connect attempt had failed, retry on timeout.
CS_DISCONNECTING	Outwards disconnect in progress.
CS_CONNECTING	Connect attempt in progress.
CS_DATA_XFER	Connected, ASP_DOWN.
CS_WAIT_ASPSM_UP_ACK	ASP UP sent, waiting for ack.
CS_WAIT_ASPSM_DOWN_ACK	ASP DOWN sent, waiting for ack.
CS_ACTIVE	ASP UP Ack sent/received
The states below are all transient, the connection returns to CS_ACTIVE when the relevant response is received, only one type of request can be outstanding at any time.	
CS_WAIT_MGMT_NOTIFY	Waiting for a MGMT NTFY following receipt of ASP UP Ack so that its routing context can be used in all further messages.
CS_WAIT_RKM_REG_RSP	Registration in progress.
CS_WAIT_RKM_DEREG_RSP	Deregistration in progress
CS_WAIT_ASPTM_ACTIVATE_ACK	Route context activation in progress.
CS_WAIT_ASPTM_INACTIVATE_ACK	Route context deactivation in progress.

The symbolic constants aren't automatically generated, but can be generated by an appropriate expansion of ACU_M3UA_CON_STATES().

B.6.4 M3UA Connection Statistics:

ACU_SS7_AMRC_TIME	Time since connection created, for outward connections the values are not reset if the connection disconnects and reconnects
ACU_SS7_AMRC_TX_MSGS	Number of transmitted data messages.
ACU_SS7_AMRC_TX_BYTES	Transmit bytes (including 12 bytes for the routing label information).
ACU_SS7_AMRC_RX_MSGS	Received data messages.
ACU_SS7_AMRC_RX_BYTES	Received bytes.
ACU_SS7_AMRC_RX_DISCARDS	Received data discarded; includes the case where the remote pointcode and si are unexpected.
ACU_SS7_AMRC_RX_DISCARD_BYTES	Discarded bytes.

Receive data is counted to either RX_MSGS or RX_DISCARDS.

B.6.5 M3UA Route Key Statistics:

ACU_SS7_AMRC_TIME	Time since route key created.
ACU_SS7_AMRC_TX_MSGS	Number of transmitted pdus.
ACU_SS7_AMRC_TX_DISCARDS	Discards because no active connection.
ACU_SS7_AMRC_RX_MSGS	Received data messages.
ACU_SS7_AMRC_RX_DISCARDS	Discards, data received but route key inactive.

Appendix C: SS7 Events

C.1 Common events

C.1.1 Server creation and deletion

These events are generated during 'ss7maint start' and driver unload for all servers. ***ame_event_bit*** will be set to ACU_SS7_EV_SERVER_STATE:

ame_state will be one of:

ACU_SS7_EVS_SERVER_DELETE	Server deleted.
ACU_SS7_EVS_SERVER_CREATE	Server created.
ACU_SS7_EVS_SERVER_CONFIG	Configuration complete.

C.2 MTP3 events

C.2.1 Destination state change

If the accessibility of a remote pointcode changes, an MTP3 event is generated that has:

ame_event_bit	ACU_SS7_EV_MTP3_DEST_STATE.
ame_remote_pc	Remote pointcode.
ame_state	MTP destination state (see B.3.1)

C.2.2 Route state change

If the state of a route changes, an MTP3 event is generated that has:

ame_event_bit	ACU_SS7_EV_MTP3_ROUTE_STATE.
ame_remote_pc	Remote pointcode.
ame_item	Adjacent pointcode.
ame_state	MTP signalling route state (see B.2.1).

C.2.3 Signalling link state change

If the state of a signalling link changes, an MTP3 event is generated that has:

ame_event_bit	ACU_SS7_EV_MTP3_LINK_STATE.
ame_remote_pc	Remote pointcode.
ame_item	slc value of the link.
ame_state	MTP Level 3 signalling link problem code, (see B.1.3)

C.3 ISUP events

ISUP events generated on distributed ISUP systems are passed to the MTP3 system(s) and reported there, as well as on the ISUP system itself.

Events from distributed ISUP systems have the IP address of the distributed system written to the `ame_system_ip` field.

C.3.1 Circuit registration events

ISUP generates an event whenever a CIC is added or removed. This happens during firmware download, or when a distributed ISUP system connects.

`ame_event_bit`	`ACU_SS7_EV_ISUP_CIC_REGISTER`.
`ame_remote_pc`	Remote pointcode.
`ame_item`	CIC number of the ISUP circuit.
`ame_state`	See table below.

Circuit registration states:

`ACU_SS7_EVS_CIC_REG_LOCAL`	Local circuit, local MTP3.
`ACU_SS7_EVS_CIC_REG_DUAL`	Circuit on dual MTP3 system.
`ACU_SS7_EVS_CIC_REG_HOST_A`	Local circuit, MTP3 on host-A.
`ACU_SS7_EVS_CIC_REG_HOST_B`	Local circuit, MTP3 on host-B.
`ACU_SS7_EVS_CIC_REG_DISTRIB`	Circuit on distributed ISUP system.
`ACU_SS7_EVS_CIC_REG_AWOL`	TCP/IP disconnection caused dual/distributed CIC to be deregistered.

C.3.2 Circuit connection state events

This event is generated whenever the call state of an ISUP circuit changes (eg when a ISUP IAM message is received).

`ame_event_bit`	`ACU_SS7_EV_ISUP_CIC_STATE`.
`ame_remote_pc`	Remote pointcode.
`ame_item`	CIC number of the ISUP circuit.
`ame_state`	ISUP circuit state from B.4.1 (non failure states only)

C.3.3 Circuit blocking events

This event is generated whenever the blocking state of an ISUP circuit changes. Group blocking will generate an event for each affected circuit.

`ame_event_bit`	`ACU_SS7_EV_ISUP_CIC_BLOCK`.
`ame_remote_pc`	Remote pointcode.
`ame_item`	CIC number of the ISUP circuit.
`ame_state`	ISUP circuit blocking problem codes from B.4.2

C.3.4 Circuit reset events

This event is generated when an ISUP circuit is reset.

`ame_event_bit`	`ACU_SS7_EV_ISUP_CIC_RESET`.
`ame_remote_pc`	Remote pointcode.
`ame_item`	CIC number of the ISUP circuit.
`ame_state`	ISUP circuit reset problem codes from B.4.2

C.4 TCAP events

C.4.1 Application registration events

TCAP generates the following event when a TCAP application connects to the driver, or enables the receipt of inward transactions. The event could be used to activate M3UA routing keys.

ame_event_bit	ACU_SS7_EV_TCAP_REGISTER.
ame_remote_ip	IP address of application system.
ame_item	SCCP subsystem number (ssn).
ame_item_1	TCAP transaction id block (masked with 0xfff00000).
ame_state	TCAP application state from Appendix B.5.1.

C.5 SIGTRAN M3UA events

C.5.1 M3UA AS-State_Change events

This event is generated whenever the overall state of an M3UA routing key changes, i.e. when a management notify message with Status Type AS-State_Change is received or sent (or would be sent for clients).

ame_event_bit	ACU_SS7_EV_M3UA_RK_STATE_CHANGE.
ame_remote_ip	IP address of M3UA peer system.
ame_item	Routing context (from local configuration).
ame_item_1	ASP identifier from remote system, ~0u if not received.
ame_state	AS state value from Appendix B.6.1.

C.5.2 M3UA connection routing context events

This event is generated whenever the state of an M3UA routing context on an SCTP connection changes.

ame_event_bit	ACU_SS7_EV_M3UA_CON_ROUTE_CTX.
ame_remote_ip	IP address of M3UA peer system.
ame_item	Routing context (from local configuration).
ame_item_1	ASP identifier from remote system, ~0u if not received.
ame_state	M3UA routing key state from Appendix B.6.2.

Appendix D: ss7maint_api.h

This header file contains all the definitions for the SS7 maintenance API.

All the definitions start `acu_ss7`, or `ACU_SS7` in order to avoid polluting other namespaces.

The definitions are all in C, but can be used from C++ applications.

Note **A significant amount of pre-processor 'magic' is used to avoid replicating information.**

D.1 Error Codes

`ACU_SS7MAINTAPI_ERROR_UNSPECIFIED`
: Unspecified error

`ACU_SS7MAINTAPI_ERROR_TIMEDOUT`
: The timeout expired before any information was available.

`ACU_SS7MAINTAPI_ERROR_BLOCK_INPROG`
: The request could not be completed because a blocking or unblocking request is currently in progress.

`ACU_SS7MAINTAPI_ERROR_RESET_INPROG`
: The request could not be completed because a reset request is currently in progress.

`ACU_SS7MAINTAPI_ERROR_BAD_STATE`
: The request could not be completed because the target object in the SS7 driver is in the wrong state for request.

`ACU_SS7MAINTAPI_ERROR_NOTFOUND`
: The requested search object could not be found.

`ACU_SS7MAINTAPI_ERROR_CONGESTION`
: The request failed due to congestion in the SS7 driver.

`ACU_SS7MAINTAPI_ERROR_NOTSUPPORTED`
: The action requested is not supported.

`ACU_SS7MAINTAPI_ERROR_TOO_MANY`
: Too many objects for action requested. Reduce the search criteria.

`ACU_SS7MAINTAPI_ERROR_CONTEXT_INUSE`
: The context supplied in the `acu_ss7maintapi_ctx_t` parameter is already in use.

`ACU_SS7MAINTAPI_ERROR_ARGUMENT_RANGE`
: One or more values supplied are outside the valid range.

`ACU_SS7MAINTAPI_ERROR_INVALID_EVENT`
: The `acu_ss7maintapi_os_event_t` parameter does not reference a valid event structure.

`ACU_SS7MAINTAPI_ERROR_INVALID_SEARCH`
: The `acu_ss7maintapi_search_t` parameter does not reference a valid search structure.

`ACU_SS7MAINTAPI_ERROR_INVALID_RESULT`
: The `acu_ss7maintapi_result_t` parameter does not reference a valid result structure.

`ACU_SS7MAINTAPI_ERROR_INVALID_CONTEXT`
: The `acu_ss7maintapi_ctx_t` parameter does not reference a valid context structure.

`ACU_SS7MAINTAPI_ERROR_NO_DRIVER`
: The library was unable to open SS7 driver.

`ACU_SS7MAINTAPI_ERROR_MALLOC_FAIL`
: The library failed to allocate memory for an item.

`ACU_SS7MAINTAPI_ERROR_SUCCESS`
: Success (guaranteed to be zero)